MAGNOLIA: Matching Algorithms via GNNs for Online Value-to-go Approximation

Alexandre Hayderi [1] **Amin Saberi** [1] **Ellen Vitercik** [1] **Anders Wikum** [1]

Abstract

Online Bayesian bipartite matching is a central problem in digital marketplaces and exchanges, including advertising, crowdsourcing, ridesharing, and kidney donor matching. We introduce a graph neural network (GNN) approach that acts as a continuous approximation to the intractable optimal online algorithm, which selects actions (e.g., which nodes to match) by computing each action's *value-to-go (VTG)*—the expected weight of the matching if the algorithm takes that action, then acts optimally in the future. We train a GNN to estimate VTG and show empirically that our method returns high-weight matchings across a variety of tasks. Moreover, we identify a common family of graph distributions in spatial crowdsourcing applications, such as rideshare, under which VTG can be efficiently approximated by aggregating information locally within graphs. This structure matches the local behavior of GNNs, providing theoretical justification for our approach.

1. Introduction

Online matching is a central problem in many digital marketplaces. In advertising, website visitors are matched to ads [Mehta et al., 2013], and on crowdsourcing platforms, crowdworkers are matched to appropriate tasks [Tong et al., 2020]. The rideshare industry faces the complex task of matching riders with drivers [Zhao et al., 2019], while in medical fields such as kidney exchange, donors must be efficiently matched to patients [Ezra et al., 2020]. The challenge is that irrevocable matching decisions must be made online without precise knowledge of how demand will evolve.

In the *Online Bayesian Bipartite Matching (OBBM)* problem, there is a weighted bipartite graph with *online* and *offline* nodes. Matching occurs in rounds, with one online node arriving with known probability in each round. The goal is to compute a high-weight matching despite not knowing which online nodes will arrive *a priori*.

The online optimal algorithm for OBBM, OPT_{on}, only has access to the distribution over online node arrivals. Upon the arrival of an online node, OPT_{on} takes the action (i.e., the choice of the matching edge or the decision to skip) that maximizes the weight of the final matching in expectation over future arrivals. This can be formulated as an exponential-sized dynamic programming routine which, in each timestep, computes the *value-to-go (VTG)* of each action—the expected final matching weight if OPT_{on} takes that action and then acts optimally in the future.

1.1. Our contributions

We present a continuous approximation to OPT_{on} that uses a graph neural network (GNN) to estimate the VTG of actions. Moreover, we provide theoretical guarantees that justify this architecture's suitability for the OBBM problem.

Key challenges. The primary obstacle we face is the sheer complexity of OBBM: for some constant $\alpha < 1$, it is PSPACE-hard to approximate the value of OPT_{on} within a factor of α [Papadimitriou et al., 2021]. Nonetheless, our approach competes with OPT_{on} across many tasks empirically, and while deep learning architectures are notoriously difficult to analyze, we prove a correspondence between the functions computable by GNNs and the VTG function.

Theoretical guarantees. We prove that VTG can be efficiently approximated by a "local" function in *bipartite random geometric graphs (b-RGGs)*. Edges in b-RGGs rely on node similarity within a latent space, so b-RGGs often arise in spatial crowdsourcing tasks like ridesharing, which necessitates physical proximity between drivers and riders.

Empirical analysis. We present MAGNOLIA, a GNN-based framework for online matching that mimics OPT_{on} by using predictions of VTG. While MAGNOLIA is only trained on small graphs for which VTG is computable, it still beats state-of-the-art baselines across a range of inputs.

[1]Stanford University, CA, USA. Correspondence to: Anders Wikum <wikum@stanford.edu>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

1.2. Related work

Online Bayesian bipartite matching. OBBM is connected to the literature on online Bayesian selection problems, where a seminal result by Krengel and Sucheston [1978] implies that no algorithm can provide better than a 0.5-approximation to the offline optimal in hindsight. However, better approximation ratios are possible when competing with the optimal *online* algorithm OPT_{on} [Papadimitriou et al., 2021, Braverman et al., 2022, Naor et al., 2023].

ML for combinatorial optimization. There has been significant recent interest in integrating ML with combinatorial optimization (see, e.g., the survey by Bengio et al. [2021]). Applications of ML to online NP-hard problems have primarily aimed to learn algorithms with good worst-case guarantees e.g. [Kong et al., 2018, Zuzic et al., 2020, Du et al., 2022]. Recent work by Alomrani et al. [2022] has considered average-case performance but is entirely empirical.

2. Notation and background

Let $G = (L, R, E)$ be a bipartite graph on a set L of n *offline nodes* and a set $R = \{1, \ldots, m\}$ of m *online nodes*, with undirected edges $E \subseteq L \times R$. Further, let $N = m + n$ and $\mathcal{N}_G(t)$ denote the neighbors of online node t in G.

2.1. Online Bayesian bipartite matching (OBBM)

An input to the OBBM problem is a bipartite graph G attributed with edge weights $\{w_{ij}\}_{(i,j)\in E}$ and online node arrival probabilities $\{p_t\}_{t\in R}$. Matching in G occurs over m timesteps. At time $t \in [m]$, online node t appears independently with probability p_t. If node t appears, one must decide to match t with an unmatched neighbor or skip t. The goal is to maximize the weight of the final matching.

An algorithm for OBBM knows the input graph G but does not know *a priori* which online nodes will arrive. Thus, in timestep t, if online node t arrives and $S \subseteq L$ is the set of offline nodes that have not yet been matched, the algorithm's choice of which node to match t to—or whether to skip t—is a function of S, t, and G. The *optimal online algorithm* OPT_{on} computes the *value-to-go function* $\mathcal{V}_G(S, t)$ to inform its decisions. This is the maximum expected matching weight achievable in G over arrivals $\{t, \ldots, m\}$, with matchings restricted to the set of remaining offline nodes S. Additional matches require available nodes, so $\mathcal{V}_G(\emptyset, t) = 0$ for all $t \in R$ and $\mathcal{V}_G(S, m + 1) = 0$ for all $S \subseteq L$. Further, the values of $\mathcal{V}_G(\cdot)$ are related by the recurrence

$$\mathcal{V}_G(S,t) = (1-\boldsymbol{p}_t)\cdot\mathcal{V}_G(S,t+1) + \boldsymbol{p}_t\cdot\max\left\{\mathcal{V}_G(S,t+1), \max_{u\in\mathcal{N}_G(t)\cap S}\mathcal{V}_G(S,t,u)\right\}$$

where $\mathcal{V}_G(S, t, u) = w_{tu} + \mathcal{V}_G(S \setminus \{u\}, t + 1)$ is the utility of adding the edge (t, u) to the matching and making u unavailable. We will refer to $\mathcal{V}(G) := \mathcal{V}_G(L(G), 1)$ as the full value-to-go computation on the input graph G.

2.2. Graph neural networks

Let $G = (V, E, \mathcal{X})$ be a graph with node attributes $\boldsymbol{h}_v^{(0)} \in \mathcal{X}$. A *Graph Neural Network* (GNN) of depth k iteratively computes a sequence of embeddings $\boldsymbol{h}_v^{(1)}, \ldots, \boldsymbol{h}_v^{(k)}$ for each node $v \in V$. In layer i, the GNN first computes a message $\boldsymbol{m}_v^{(i)}$ for each node v from its previous embedding $\boldsymbol{h}_v^{(i-1)}$. Then, the next embedding $\boldsymbol{h}_v^{(i)}$ is computed by aggregating the messages $\boldsymbol{m}_u^{(i)}$ from each of v's neighbors:

$$\boldsymbol{m}_v^{(i)} = \text{MSG}^{(i)}\left(\boldsymbol{h}_v^{(i-1)}\right) \qquad \text{for all } v \in V$$

$$\boldsymbol{h}_v^{(i)} = \text{AGGREGATE}^{(i)}\left(\boldsymbol{m}_v^{(i)}, \{\boldsymbol{m}_u^{(i)} : u \in \mathcal{N}(v)\}\right).$$

Note that a GNN can act on graphs with any number of nodes, and that the embedding $\boldsymbol{h}_v^{(k)}$ is a function only of the embeddings within a k-hop neighborhood of node v.

3. Theoretical guarantees

In this section, we identify conditions on the generating parameters of *bipartite random geometric graphs* (b-RGGs) for which VTG can be approximated by aggregating information over local neighborhoods. This result aligns with the inherent processing capabilities of GNNs.

3.1. Local graph decomposition

We begin by showing that, under certain conditions, b-RGGs admit a *local decomposition*. Informally, this means that:

1. (Decomposable) b-RGGs can be partitioned into subgraphs such that few edges cross between subgraphs.
2. (Local) Under mild assumptions, the number of nodes in each resulting subgraph is relatively small.

Bipartite random geometric graphs. In b-RGGs, nodes are connected when their embeddings are sufficiently close in some metric. Our results generalize to any p-norm.

Definition 3.1. *Given a distribution $\mathcal{D}$ over $[0, 1]^d$, a bipartite random geometric graph $G(m, n, \mathcal{D}, \Delta)$ is a distribution over graphs on m online and n offline nodes where each node has an embedding $\boldsymbol{x_i} \sim \mathcal{D}$. There is an edge between online node i and offline node j if and only if $\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_\infty \leq \Delta$.*

A partition π of $[0, 1]^d$ induces a partition of b-RGGs into subgraphs: let $G(\pi)$ be formed by removing edges (i, j) in G with embeddings $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ in different cells of π. We can map properties of the partition π to properties of $G(\pi)$.

Random k-partitions. We introduce a random partitioning scheme that splits $[0,1]^d$ into cells of equal volume and applies a random "shift" to these cells in each dimension.

Definition 3.2. *For $k \in \mathbb{Z}_{\geq 1}$, a $(k, \boldsymbol{s})$-partition of $[0,1]^d$ is the partition $\boldsymbol{s}_i + \{0, \frac{1}{k}, \dots, \frac{k-1}{k}\}$ along each dimension i, where $\boldsymbol{s}_i \in [0, \frac{1}{k}]$.*

b-RGG decomposition. Let Π_k denote a uniform distribution over $(k, \boldsymbol{s})$-partitions with $\boldsymbol{s} \sim U(0, 1/k)^d$ and call $\pi \sim \Pi_k$ a *random k-partition*. For carefully chosen k, nearby vectors are likely to lie in the same cell of π. So, edges in G are unlikely to be removed when forming $G(\pi)$.

Lemma 3.3. *Let $\boldsymbol{x_1}, \dots, \boldsymbol{x}_N \in [0,1]^d$, $\varepsilon > 0$, $\Delta \leq \frac{\varepsilon}{2d}$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$. If $\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_\infty \leq \Delta$, then with probability at most ε over $\pi \sim \Pi_k$, $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ lie in different cells of π.*

Moreover, the number of b-RGG latent embeddings in any cell of a random k-partition with $\Omega(N)$ cells is likely to be sublinear in N. We avoid the pathological case where $\mathcal{D}$ is a point mass using a concept from smoothed analysis.

Definition 3.4 (Haghtalab et al. [2022]). *A distribution over $[0,1]^d$ with density function f is β-smooth if $\sup f(\boldsymbol{x}) \leq \beta$.*

Now, treat sampling N vectors from $\mathcal{D}$ as a balls-into-bins process: balls are vectors, and bins are cells of a k-partition.

Corollary 3.5. *Suppose N vectors are sampled from a β-smooth distribution over $[0,1]^d$. For all $\pi \in \operatorname{supp}(\Pi_k)$ where $k^d = \Omega(N)$, every cell of π contains $O(\beta \log N)$ vectors with probability $1 - O(\frac{1}{N})$ for N sufficiently large.*

Lemma 3.3 and Corollary 3.5 give the following theorem.

Theorem 3.6. *Let $\mathcal{D}$ be a β-smooth distribution, $\Delta = O(N^{-1/d})$, $\varepsilon > 0$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$. Then,*

1. *(Decomposable) For any $G \in \operatorname{supp}(G(m, n, \mathcal{D}, \Delta))$, each edge $e \in E(G)$ appears in $G(\pi)$ with probability at least $1 - \varepsilon$ over the draw of $\pi \sim \Pi_k$.*
2. *(Local) For any $\pi \in \operatorname{supp}(\Pi_k)$ and N sufficiently large, the connected components of $G(\pi)$ are of size $O(\beta \log N)$ with probability $1 - O(1/N)$ over the draw of $G \sim G(m, n, \mathcal{D}, \Delta)$.*

3.2. Local approximation of value-to-go

This section shows that the local decomposability of b-RGGs means VTG can be approximated using *local graph functions*, which are computed on small substructures.

Definition 3.7 (Tahmasebi et al. [2023]). *A function f over graphs is r-local if there is a function φ such that $f(G) = \varphi(\{\mathcal{N}_r(v)\}_{v \in V(G)})$ where $\mathcal{N}_r(v)$ is the r-hop neighborhood of node v in G.*

We prove that with high probability, VTG is approximated by a $O(\beta \log N)$-local function, formalized as follows.

Definition 3.8. *A function f on graphs is (r, ε, δ)-locally approximable over a random graph family $\mathcal{G}$ if there is an r-local, polynomial time-computable function h such that $|f(G) - h(G)| \leq \varepsilon f(G)$ with probability $1 - \delta$ over $G \sim \mathcal{G}$ and any randomness in h.*

Local approximation for b-RGGs. The VTG of an OBBM instance can only decrease after removing edges to form $G(\pi)$. Moreover, because each edge of G exists in $G(\pi)$ for a $1 - \varepsilon$ fraction of random $\lceil \frac{\varepsilon}{2d\Delta} \rceil$-partitions π, $\mathcal{V}(G(\pi))$ is at least $(1 - \varepsilon)\mathcal{V}(G)$ in expectation.

Lemma 3.9. *For $G \in \operatorname{supp}(G(m, n, \mathcal{D}, \Delta))$, $\varepsilon > 0$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$, $\mathcal{V}(G) \geq \mathbb{E}_{\pi \sim \Pi_k} \left[\mathcal{V}(G(\pi)) \right] \geq (1 - \varepsilon)\mathcal{V}(G)$.*

While $\tilde{\mathcal{V}}(G) = \mathbb{E}_{\pi \sim \Pi_k} \left[\mathcal{V}(G(\pi)) \right]$ can approximate $\mathcal{V}(G)$ to high accuracy over b-RGGs, it may not be r-local for small r. This requires the connected components of $G(\pi)$ to be of size at most r under all partitions $\pi \in \operatorname{supp}(\Pi_k)$. Unlike $\tilde{\mathcal{V}}(\cdot)$, a random function that outputs a sample estimate $\frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{V}(G(\pi_i))$ achieves r-locality when the connected components of $G(\pi_1), \dots, G(\pi_\ell)$ are of size at most r.

Lemma 3.10. *Let $\mathcal{D}$ be β-smooth and $\Delta = O(N^{-1/d})$ for N sufficiently large. For $\varepsilon \in (0, 1/2]$, let $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$. With probability $1 - \delta$ over the draw of $G \sim G(m, n, \mathcal{D}, \Delta)$ and $\ell \geq \frac{2}{\varepsilon^2} \log \frac{4}{\delta}$ partitions $\pi_1, \dots, \pi_\ell \sim \Pi_k$, each $\mathcal{V}(G(\pi_i))$ can be computed by a $O(\beta \log N)$-local function and*

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{V}(G(\pi_i)) \geq (1 - \varepsilon) \mathop{\mathbb{E}}_{\pi \sim \Pi_k} [\mathcal{V}(G(\pi))].$$

Finally, we give our main theorem.

Theorem 3.11. *Given a β-smooth distribution $\mathcal{D}$ over $[0,1]^d$ and $\Delta = O(N^{-1/d})$, for sufficiently large N, the VTG function $\mathcal{V}$ is $\left(O(\beta \log N), \varepsilon, \delta\right)$-locally approximable over $G(m, n, \mathcal{D}, \Delta)$ for all $\varepsilon \in (0, \frac{1}{2}]$ and $\delta \in (0, 1]$.*

4. Experiments

Additional details on our setup can be found in Appendix B, with comprehensive experimental results in Appendix B.8.

4.1. Experimental setup

Learned matching model. MAGNOLIA replaces the VTG computations in OPT_{on} with predictions from a GNN (see Figure 1). To start, the *matching state* is encoded as an attributed graph consisting of the graph G, arriving online node t, and set of available offline nodes S. This attributed graph is fed into a GNN, which predicts the VTG associated with each feasible action. The decision with the highest predicted VTG is chosen, and the process repeats until no online nodes remain. The GNN is trained using supervised learning with teacher forcing on the decisions of OPT_{on}.

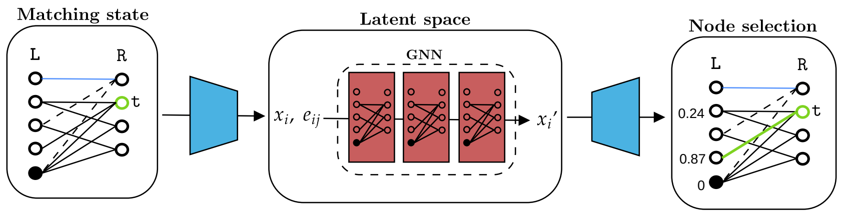

Figure 1. MAGNOLIA's GNN-based matching subroutine. Blue edges are in the matching at time t, the arriving node is highlighted green, and the green edge shows MAGNOLIA's selected action. A virtual node with zero utility (colored black) is added to allow skipping.

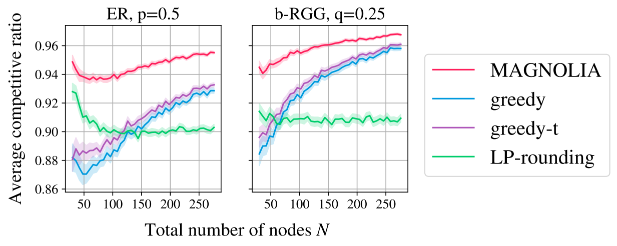

Figure 2. Evolution of average CR over ER and b-RGG graphs of increasing size with a 2:1 ratio of online to offline nodes.

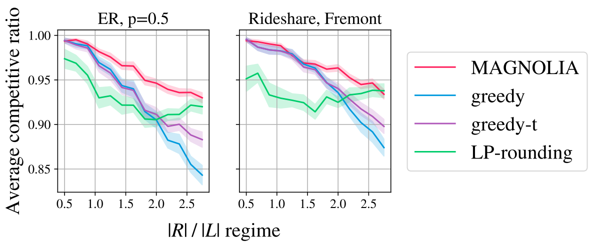

Figure 3. Average CR for different regimes of ER and OSMNX graphs. $|L| = 16$ is fixed and $|R|$ varies from 8 to 45 online nodes.

Instance generation. Training and testing are done on synthetic and semi-synthetic weighted bipartite graphs with arrival probabilities $p_t \sim U(0, 1)$. Synthetic inputs are drawn from the Erdős-Rényi (ER) [Erdős and Rényi, 1960], Barabási-Albert (BA) [Albert and Barabási, 2002], and geometric (b-RGG) random graph families. We generate semi-synthetic graphs from OSMnx [Boeing, 2017], a library that encodes road networks used in ride-sharing applications, and the gMission dataset [Chen et al., 2014], whose base graph comes from crowdsourcing data for assigning workers to tasks. We say a bipartite graph has shape $(|L|\times|R|)$ if it has $|L|$ offline nodes and $|R|$ online nodes.

Baselines. We compare MAGNOLIA to several strong baselines using average *competitive ratio (CR)*:

$$\mathrm{CR}(M) = \frac{1}{k\cdot\ell}\sum_{i=1}^{k}\sum_{j=1}^{\ell}\frac{M(G_i, \boldsymbol{a}_{ij})}{\mathrm{OPT}(G_i, \boldsymbol{a}_{ij})}.$$

$M(G, \boldsymbol{a})$ is the matching weight returned by method M on graph G with realized online node arrivals $\boldsymbol{a}$, $\mathrm{OPT}(G, \boldsymbol{a})$ is the max matching weight in G based on a priori knowledge of $\boldsymbol{a}$, and we average over k graphs with ℓ realizations of the online node arrivals per graph.

When an online node arrives, `greedy` picks the maximum weight available edge. To trade off between short and long-term rewards, `greedy-t` [Alomrani et al., 2022], makes the same decision as `greedy` if the edge weight is above some learned threshold t, and skips otherwise. Finally, `LP-rounding` is a 0.632-approximation to OPT_{on} by Braverman et al. [2022], which is strong in practice.

4.2. Results

We train MAGNOLIA's GNN on 2000 OBBM instances from 3 graph configurations, then evaluate performance on 6000 unseen instances from a broader set of 12 configurations. A key characteristic of GNNs trained for combinatorial tasks is the extent to which they generalize to graphs of larger size. Despite being trained on 16-node graphs, we see in Figure 2 that MAGNOLIA's performance remains consistent even for inputs up to 20 times larger.

Further, we observe that the ratio of online to offline nodes in an input has a big impact on the performance of OBBM algorithms. In light of this, we replace MAGNOLIA's GNN with a meta-model that selects between two GNNs—one trained on graphs of size (10×6) and the other (6×10)—on an instance-by-instance basis. Figure 3 gives generalization results for MAGNOLIA with this meta-model. It consistently outperforms greedy baselines and performs especially well for ratios where `LP-rounding` is worst.

5. Conclusions

In this paper, we studied Online Bayesian Bipartite Matching, a central problem in digital marketplaces. We introduced MAGNOLIA, a novel approximation to the intractable optimal online algorithm. Theoretically, we showed that value-to-go can be efficiently approximated in random geometric graphs using local functions—a process well-suited to GNNs. Empirically, MAGNOLIA beats state-of-the-art baselines, showing strong out-of-distribution generalization.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, page 2623–2631, 2019. B.6

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 2002. 4.1, B.2.1

Mohammad Ali Alomrani, Reza Moravej, and Elias Boutros Khalil. Deep policies for online bipartite matching: A reinforcement learning approach. *Transactions on Machine Learning Research (TMLR)*, 2022. 1.2, 4.1, B.2.2

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. 1.2

Geoff Boeing. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65: 126–139, 2017. 4.1, B.2.2

Allan Borodin, Christodoulos Karavasilis, and Denis Pankratov. An experimental study of algorithms for online bipartite matching. *Journal of Experimental Algorithmics (JEA)*, 2020. B.2.1

Mark Braverman, Mahsa Derakhshan, and Antonio Molina Lovett. Max-weight online stochastic matching: Improved approximations against the online benchmark. In *ACM Conference on Economics and Computation (EC)*, page 967–985, 2022. 1.2, 4.1

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. B.7

Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. gmission: A general spatial crowdsourcing platform. *Proceedings of the VLDB Endowment*, 7(13):1629–1632, 2014. 4.1

Bingqian Du, Zhiyi Huang, and Chuan Wu. Adversarial deep learning for online resource allocation. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 6(4):1–25, 2022. 1.2

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960. 4.1, B.2.1

Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. Online stochastic max-weight matching: prophet inequality for vertex and edge arrival models. In *ACM Conference on Economics and Computation (EC)*, pages 769–787, 2020. 1

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019. B.4

Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. Smoothed analysis with adaptive adversaries. In *Symposium on Foundations of Computer Science (FOCS)*, pages 942–953. IEEE, 2022. 3.4

Weiwei Kong, Christopher Liaw, Aranyak Mehta, and D Sivakumar. A new dog learns old tricks: RL finds classic optimization algorithms. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 1.2

Ulrich Krengel and Louis Sucheston. On semiamarts, amarts, and processes with finite value. *Probability on Banach Spaces*, 4:197–266, 1978. 1.2

Guohao Li, Chenxin Xiong, Ali K. Thabet, and Bernard Ghanem. DeeperGCN: All you need to train deeper GCNs. *CoRR*, abs/2006.07739, 2020. B.4, B.7

Aranyak Mehta et al. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013. 1

Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005. A

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2019. B.7

Joseph Naor, Aravind Srinivasan, and David Wajc. Online dependent rounding schemes. *ArXiv*, abs/2301.08680, 2023. 1.2

Christos Papadimitriou, Tristan Pollner, Amin Saberi, and David Wajc. Online stochastic max-weight bipartite matching: Beyond prophet inequalities. In *ACM Conference on Economics and Computation (EC)*, page 763–764, 2021. 1.1, 1.2

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2951–2959, 2012. B.6

Behrooz Tahmasebi, Derek Lim, and Stefanie Jegelka. The power of recursion in graph neural networks for counting substructures. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 11023–11042, 2023. 3.7

Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29:217–250, 2020. 1

Boming Zhao, Pan Xu, Yexuan Shi, Yongxin Tong, Zimu Zhou, and Yuxiang Zeng. Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 2245–2252, 2019. 1

Goran Zuzic, Di Wang, Aranyak Mehta, and D Sivakumar. Learning robust algorithms for online allocation problems using adversarial training. *arXiv preprint arXiv:2010.08418*, 2020. 1.2

A. Proofs

Lemma 3.3. *Let $\boldsymbol{x_1}, \dots, \boldsymbol{x}_N \in [0,1]^d$, $\varepsilon > 0$, $\Delta \leq \frac{\varepsilon}{2d}$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$. If $\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_\infty \leq \Delta$, then with probability at most ε over $\pi \sim \Pi_k$, $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ lie in different cells of π.*

Proof. Consider $\pi \sim \Pi_k$, and let π_ℓ be the boundaries of π along dimension ℓ for $\ell \in [d]$. For $i, j \in [N]$ such that $||\boldsymbol{x_i} - \boldsymbol{x_j}||_\infty \leq \Delta$, notice that $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ lie in different cells of π precisely when in at least one dimension ℓ, some point in π_ℓ falls in the interval $[(\boldsymbol{x_i})_\ell, (\boldsymbol{x_j})_\ell]$. By symmetry, the probability that this occurs is equal to the length of the interval $[(\boldsymbol{x_i})_\ell, (\boldsymbol{x_j})_\ell]$ over the measure of possible shifts $1/k$. Moreover, $\frac{|(\boldsymbol{x_i})_\ell - (\boldsymbol{x_j})_\ell|}{1/k} \leq k\Delta$, so

$$\begin{aligned}\Pr_{\pi\sim\Pi_k}[\boldsymbol{x_i} \text{ and } \boldsymbol{x_j} \text{ lie in different cells of } \pi] &\leq 1 - (1-k\Delta)^d \\ &\leq 1 - \left(1 - \left(\frac{\varepsilon}{2d\Delta} + 1\right)\Delta\right)^d \\ &\leq 1 - \left(1 - \frac{\varepsilon}{d}\right)^d \\ &\leq 1 - (1-\varepsilon) \\ &= \varepsilon.\end{aligned}$$

□

Lemma A.1. *For $\beta \geq 1$, when N balls are dropped independently into $K = \Omega(N)$ bins and the probability a particular ball lands in each bin is at most $\frac{\beta}{K}$, the probability the maximum load is more than $\frac{3\beta \ln N}{\ln\ln N}$ is $O(\frac{1}{N})$ for N sufficiently large.*

Proof. Let c be a constant such that $K \geq cN$ for N sufficiently large. Following Lemma 5.1 from [Mitzenmacher and Upfal, 2005], the probability that at least M balls are dropped in bin 1 is at most

$$\binom{N}{M}\left(\frac{\beta}{K}\right)^M$$

by a union bound over the probability of each subset of M balls being dropped in bin 1. In particular, there are $\binom{N}{M}$ possible subsets of balls, and the probability of selecting bin 1 for each of M chosen balls is bounded above by $\left(\beta/K\right)^M$. By another union bound over the K bins, the probability that any bin has a load of at least M balls is at most

$$K\binom{N}{M}\left(\frac{\beta}{K}\right)^M.$$

We use the fact that $f(x) = x\binom{N}{M}\left(\frac{\beta}{x}\right)^M$ is decreasing for $x > 0$ and the inequalities

$$\binom{N}{M}\left(\frac{1}{N}\right)^M \leq \frac{1}{M!} \leq \left(\frac{e}{M}\right)^M$$

to conclude that

$$K\binom{N}{M}\left(\frac{\beta}{K}\right)^M \leq cN\binom{N}{M}\left(\frac{\beta}{cN}\right)^M = cN\left(\frac{\beta}{c}\right)^M \cdot \binom{N}{M}\left(\frac{1}{N}\right)^M \leq cN\left(\frac{\beta e}{cM}\right)^M.$$

For $M \geq 3\beta \ln N/\ln\ln N$, the probability that any bin receives more than M balls in bounded above by

$$
\begin{aligned}
cN\left(\frac{\beta e}{cM}\right)^M &\leq cN\left(\frac{e\ln\ln N}{3c\ln N}\right)^{3\beta\ln N/\ln\ln N} \\
&\leq cN\left(\frac{\ln\ln N}{c\ln N}\right)^{3\beta\ln N/\ln\ln N} \\
&= e^{\ln c+\ln N}\left(e^{\ln\ln\ln N-\ln\ln N-\ln c}\right)^{3\beta\ln N/\ln\ln N} \\
&= e^{\ln c+(1-3\beta)\ln N+3\beta\ln N\left(\frac{\ln\ln\ln N-\ln c}{\ln\ln N}\right)} \\
&\leq ce^{-2\ln N+3\beta\ln N\left(\frac{\ln\ln\ln N-\ln c}{\ln\ln N}\right)} \\
&\leq \frac{c}{N} \\
&= O\left(1/N\right)
\end{aligned}
$$

for N sufficiently large. $\square$

Corollary 3.5. *Suppose N vectors are sampled from a β-smooth distribution over $[0,1]^d$. For all $\pi \in \text{supp}(\Pi_k)$ where $k^d = \Omega(N)$, every cell of π contains $O(\beta \log N)$ vectors with probability $1 - O(\frac{1}{N})$ for N sufficiently large.*

Proof. There are $k^d = \Omega(N)$ cells of π, each of volume $\frac{1}{k^d}$. Since $\mathcal{D}$ is β-smooth, the total density of $\mathcal{D}$ in each cell is at most $\frac{\beta}{k^d}$. Treating the sampling of vectors from $\mathcal{D}$ as a balls-into-bins process where balls are the N vectors, and bins are the $\Omega(N)$ cells of π, the result follows from Lemma A. We note that $\frac{\log N}{\log\log N} = O(\log N)$. $\square$

Theorem 3.6. *Let $\mathcal{D}$ be a β-smooth distribution, $\Delta = O(N^{-1/d})$, $\varepsilon > 0$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$. Then,*

1. *(Decomposable) For any $G \in \text{supp}(G(m,n,\mathcal{D},\Delta))$, each edge $e \in E(G)$ appears in $G(\pi)$ with probability at least $1-\varepsilon$ over the draw of $\pi \sim \Pi_k$.*

2. *(Local) For any $\pi \in \text{supp}(\Pi_k)$ and N sufficiently large, the connected components of $G(\pi)$ are of size $O(\beta \log N)$ with probability $1 - O(1/N)$ over the draw of $G \sim G(m,n,\mathcal{D},\Delta)$.*

Proof. (1) The definition of a b-RGG ensures that for $G \in \text{support}(G(m,n,\mathcal{D},\Delta))$ an edge (i,j) can only exist if $||\boldsymbol{x_i} - \boldsymbol{x_j}||_\infty \leq \Delta$. By Lemma 3.3, $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ belong to the same cell of $\pi \sim \Pi_k$ with probability at least $1-\varepsilon$. Equivalently, i and j belong to the same subgraph of $G(\pi)$ with probability at least $1-\varepsilon$.

(2) Corollary 3.5 implies that for $\pi \in \text{support}(\Pi_k)$, the maximum number of latent embeddings of $G \sim G(m,n,\mathcal{D},\Delta)$ in any cell of π is $O(\beta \log N)$ with probability at least $1 - O(1/N)$. The result follows from the observation that nodes in the same subgraph of $G(\pi)$ must have latent embeddings in the same cell of π. $\square$

Lemma 3.9. *For $G \in \text{supp}(G(m,n,\mathcal{D},\Delta))$, $\varepsilon > 0$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$, $\mathcal{V}(G) \geq \mathbb{E}_{\pi\sim\Pi_k}\left[\mathcal{V}(G(\pi))\right] \geq (1-\varepsilon)\mathcal{V}(G)$.*

Proof. For the upper bound, we will use an inductive argument on the number of online nodes in G. It is clear that $\mathcal{V}(G(\pi)) = \mathcal{V}(G) = 0$ for any partition π when G has no online nodes since there is nothing to match. Now, assume that $\mathcal{V}(G(\pi)) \leq \mathcal{V}(G)$ for all graphs $G \in \text{support}(G(m,n,\mathcal{D},\Delta))$ with at most $t-1$ online nodes for some $t \geq 1$ and for all partitions π. Let $G \in \text{support}(G(m,n,\mathcal{D},\Delta))$ be a graph on t online nodes and let π be any hypercube partition. Then,

$$\begin{aligned}
\mathcal{V}(G) &= \mathcal{V}_G(L,1)\\
&= (1-\boldsymbol{p}_1)\cdot \mathcal{V}_G(L,2) + \boldsymbol{p}_1 \cdot \max\left\{\mathcal{V}_G(L,2), \max_{u\in\mathcal{N}_G(1)} \{w_{1u} + \mathcal{V}_G(L\setminus\{u\},2)\}\right\}\\
&\geq (1-\boldsymbol{p}_1)\cdot \mathcal{V}_{G(\pi)}(L,2) + \boldsymbol{p}_1 \cdot \max\left\{\mathcal{V}_{G(\pi)}(L,2), \max_{u\in\mathcal{N}_G(1)} \{w_{1u} + \mathcal{V}_{G(\pi)}(L\setminus\{u\},2)\}\right\}\\
&\geq (1-\boldsymbol{p}_1)\cdot \mathcal{V}_{G(\pi)}(L,2) + \boldsymbol{p}_1 \cdot \max\left\{\mathcal{V}_{G(\pi)}(L,2), \max_{u\in\mathcal{N}_{G(\pi)}(1)} \{w_{1u} + \mathcal{V}_{G(\pi)}(L\setminus\{u\},2)\}\right\}\\
&= \mathcal{V}_{G(\pi)}(L,1)\\
&= \mathcal{V}(G(\pi)).
\end{aligned}$$

The first inequality is an application of the inductive hypothesis, since $\mathcal{V}_G(L,2)$ and $\mathcal{V}_G(L\setminus\{u\},2)$ are both full value-to-go computations on a subgraph of G with $t-1$ nodes. The second follows from the fact that $\mathcal{N}_{G(\pi)}(1) \subseteq \mathcal{N}_G(1)$, as $G(\pi)$ is formed from G by removing edges.

For the lower bound, it is helpful to first decompose the value-to-go $\mathcal{V}(G)$ into a contribution from each edge $e \in E(G)$. To do so, we make use of the fact that $\mathcal{V}(G)$ is the expected value of the matching returned by OPT$_{on}$. In greater detail, let $\boldsymbol{a} \in \{0,1\}^m$ represent an arrival sequence of online nodes where node t arrives if $\boldsymbol{a}_t = 1$ and does not arrive if $\boldsymbol{a}_t = 0$. The likelihood of observing different arrival sequences is governed by the arrival probability vector $\boldsymbol{p}$. Namely, for $\boldsymbol{a} \in \{0,1\}^m$,

$$\Pr[\boldsymbol{a}] = \prod_{t=1}^{m} \big(\boldsymbol{p}_t \cdot \boldsymbol{a}_t + (1-\boldsymbol{p}_t)\cdot(1-\boldsymbol{a}_t)\big).$$

Notice that all randomness in the output of OPT$_{on}$ comes from the random arrivals, so given a fixed arrival sequence $\boldsymbol{a}$, OPT$_{on}$ returns a deterministic matching $M(\boldsymbol{a})$. Then, we can write

$$\mathcal{V}(G) = \sum_{\boldsymbol{a}\in\{0,1\}^m} \left(\Pr[\boldsymbol{a}] \cdot \sum_{e\in M(\boldsymbol{a})} w_e\right) = \sum_{e\in E(G)} w_e \cdot \left(\sum_{\boldsymbol{a}\in\{0,1\}^m \,:\, e\in M(\boldsymbol{a})} \Pr[\boldsymbol{a}]\right) = \sum_{e\in E(G)} \alpha_e w_e,$$

where

$$\alpha_e = \sum_{\boldsymbol{a}\in\{0,1\}^m \,:\, e\in M(\boldsymbol{a})} \Pr[\boldsymbol{a}].$$

Crucially, notice that for any partition π,

$$\mathcal{V}(G(\pi)) \geq \sum_{e\in E(G)} \alpha_e w_e \cdot 1\{e \in E(G(\pi))\}.$$

The right-hand side is the expected value of the matching returned by an online algorithm on $G(\pi)$ which, for any arrival sequence $\boldsymbol{a}$, outputs $M(\boldsymbol{a}) \cap E(G(\pi))$. The left-hand side is the expected value of the matching returned by OPT$_{on}$ on $G(\pi)$. It follows immediately from these facts and Lemma 3.3 that

$$\mathop{\mathbb{E}}_{\pi\sim\Pi_k}\big[\mathcal{V}(G(\pi))\big] \geq \mathop{\mathbb{E}}_{\pi\sim\Pi_k}\left[\sum_{e\in E(G)} \alpha_e w_e \cdot 1\{e\in E(G(\pi))\}\right] \geq (1-\varepsilon) \sum_{e\in E(G)} \alpha_e w_e = (1-\varepsilon)\cdot\mathcal{V}(G).$$

□

Lemma 3.10. *Let $\mathcal{D}$ be β-smooth and $\Delta = O(N^{-1/d})$ for N sufficiently large. For $\varepsilon \in (0, 1/2]$, let $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$. With probability $1-\delta$ over the draw of $G \sim G(m, n, \mathcal{D}, \Delta)$ and $\ell \geq \frac{2}{\varepsilon^2} \log \frac{4}{\delta}$ partitions $\pi_1, \ldots, \pi_\ell \sim \Pi_k$, each $\mathcal{V}(G(\pi_i))$ can be computed by a $O(\beta \log N)$-local function and*

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{V}(G(\pi_i)) \geq (1-\varepsilon) \mathop{\mathbb{E}}_{\pi \sim \Pi_k} [\mathcal{V}(G(\pi))].$$

Proof. To simplify notation, we refer to the sample mean $\frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{V}(G(\pi_i))$ and true mean $\mathop{\mathbb{E}}_{\pi \sim \Pi_k} [\mathcal{V}(G(\pi))]$ as S_ℓ and $\mathbb{E}[S_\ell]$, respectively. Also let $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_\ell\}$ be shorthand for an i.i.d. sample of ℓ partitions from Π_k, and let $\mathcal{G}$ be shorthand for $G(m, n, \mathcal{D}, \Delta)$.

Consider the following events. For $G \in \text{support}(\mathcal{G})$ and $\boldsymbol{\pi} \in \text{support}(\Pi_k)$,

- $A(G, \boldsymbol{\pi})$ is the event that the approximation $(1-\varepsilon) \cdot \mathbb{E}[S_\ell] \leq S_\ell \leq (1+\varepsilon) \cdot \mathbb{E}[S_\ell]$ holds on G for partitions $\boldsymbol{\pi}$.
- $B(G, \boldsymbol{\pi})$ is the event the connected components of $G(\pi_i)$ are of size $O(\beta \log N)$ for each partition $\pi_i \in \boldsymbol{\pi}$. When this is the case, $\mathcal{V}(G(\pi_i))$ can be computed exactly by a $O(\beta \log N)$-local function that simply computes VTG over a $O(\beta \log N)$-hop neighborhood.

We need to show that for N sufficiently large, the event $A(G, \boldsymbol{\pi}) \wedge B(G, \boldsymbol{\pi})$ occurs with probability at least $1-\delta$ over the random draws of $G \sim \mathcal{G}$ and $\boldsymbol{\pi} \sim \Pi_k$. Toward that end, notice that

$$\begin{aligned}
\Pr_{G \sim \mathcal{G}, \boldsymbol{\pi} \sim \Pi_k} \left[A(G, \boldsymbol{\pi}) \wedge B(G, \boldsymbol{\pi})\right] &= 1 - \Pr_{G \sim \mathcal{G}, \boldsymbol{\pi} \sim \Pi_k} \left[A(G, \boldsymbol{\pi})^{\complement} \vee B(G, \boldsymbol{\pi})^{\complement}\right] \\
&\geq 1 - \Pr_{G \sim \mathcal{G}, \boldsymbol{\pi} \sim \Pi_k} \left[A(G, \boldsymbol{\pi})^{\complement}\right] - \Pr_{G \sim \mathcal{G}, \boldsymbol{\pi} \sim \Pi_k} \left[B(G, \boldsymbol{\pi})^{\complement}\right].
\end{aligned}$$

We have from the tower property of conditional expectation that

$$\Pr_{G \sim \mathcal{G}, \boldsymbol{\pi} \sim \Pi_k} \left[A(G, \boldsymbol{\pi})^{\complement}\right] = \mathop{\mathbb{E}}_{G \sim \mathcal{G}} \left[\Pr_{\boldsymbol{\pi} \sim \Pi_k} [A(G, \boldsymbol{\pi})^{\complement} \mid G]\right]$$

and

$$\Pr_{G \sim \mathcal{G}, \boldsymbol{\pi} \sim \Pi_k} \left[B(G, \boldsymbol{\pi})^{\complement}\right] = \mathop{\mathbb{E}}_{\boldsymbol{\pi} \sim \Pi_k} \left[\Pr_{G \sim \mathcal{G}} [B(G, \boldsymbol{\pi})^{\complement} \mid \boldsymbol{\pi}]\right].$$

By Theorem 3.6 and a union bound over the ℓ drawn partitions, we have that

$$\Pr_{G \sim \mathcal{G}} \left[B(G, \boldsymbol{\pi})^{\complement} \mid \boldsymbol{\pi}\right] \leq O(\ell/N) \leq \delta/2$$

for N sufficiently large.

To bound $\Pr_{\boldsymbol{\pi} \sim \Pi_k} [A(G, \boldsymbol{\pi})^{\complement} \mid G]$, first notice that for fixed $G \in \text{support}(\mathcal{G})$ and $\boldsymbol{\pi} \sim \Pi_k$, the $\mathcal{V}(G(\pi_i))$'s are i.i.d. random variables which take values in the interval $[0, \mathcal{V}(G)]$ by Lemma 3.9. Applying a standard Hoeffding bound, for $\ell \geq \frac{2}{\varepsilon^2} \log(\frac{4}{\delta})$ sampled partitions the probability of a bad approximation is

$$\begin{aligned}
\Pr\left[\left|S_\ell - \mathbb{E}[S_\ell]\right| \geq \varepsilon \mathbb{E}[S_\ell]\right] &\leq \Pr\left[\left|S_\ell - \mathbb{E}[S_\ell]\right| \geq \varepsilon(1-\varepsilon)\mathcal{V}(G)\right] && \text{Lemma 3.9} \\
&\leq 2\exp\left(-\frac{2\varepsilon^2(1-\varepsilon)^2 \mathcal{V}(G)^2}{\ell \cdot \mathcal{V}(G)^2/\ell^2}\right) \\
&\leq 2\exp\left(-\ell\varepsilon^2/2\right) \\
&\leq \delta/2.
\end{aligned}$$

Thus for sufficiently large N, we've shown that

$$\begin{aligned}\Pr_{G\sim\mathcal{G},\boldsymbol{\pi}\sim\Pi_k}[A(G,\boldsymbol{\pi})\wedge B(G,\boldsymbol{\pi})] &\geq 1-\mathop{\mathbb{E}}_{G\sim\mathcal{G}}\left[\Pr_{\boldsymbol{\pi}\sim\Pi_k}[A(G,\boldsymbol{\pi})^{\complement}\mid G]\right]-\mathop{\mathbb{E}}_{\boldsymbol{\pi}\sim\Pi_k}\left[\Pr_{G\sim\mathcal{G}}[B(G,\boldsymbol{\pi})^{\complement}\mid \boldsymbol{\pi}]\right]\\ &\geq 1-\mathop{\mathbb{E}}_{G\sim\mathcal{G}}[\delta/2]-\mathop{\mathbb{E}}_{\boldsymbol{\pi}\sim\Pi_k}[\delta/2]\\ &=1-\delta.\end{aligned}$$

□

Theorem 3.11. *Given a β-smooth distribution $\mathcal{D}$ over $[0,1]^d$ and $\Delta = O(N^{-1/d})$, for sufficiently large N, the VTG function $\mathcal{V}$ is $\big(O(\beta\log N\big), \varepsilon, \delta)$-locally approximable over $G(m,n,\mathcal{D},\Delta)$ for all $\varepsilon \in (0,\frac{1}{2}]$ and $\delta\in(0,1]$.*

Proof. Let $k = \lceil\frac{\varepsilon}{2d\Delta}\rceil$ and let $\varepsilon' = 1-\sqrt{1-\varepsilon}$ so that $(1-\varepsilon')^2 = 1-\varepsilon$. By Lemma 3.9 with $\varepsilon = \varepsilon'$, we have that $\mathbb{E}_{\pi\sim\Pi_k}[\mathcal{V}(G(\pi))] \geq (1-\varepsilon')\cdot\mathcal{V}(G)$ for all $G\in\operatorname{supp}(G(m,n,\mathcal{D},\Delta)$. Now, consider the random function $h(G)$ which samples $\ell = \frac{2}{\varepsilon'^2}\log(\frac{4}{\delta})$ partitions $\pi_1,\ldots,\pi_\ell$ from Π_k then outputs $\frac{1}{|I|}\sum_{i\in I}\mathcal{V}(G(\pi_i))$, where $I\subseteq[\ell]$ is the set of indices for which $\mathcal{V}(G(\pi_i))$ is $O(\beta\log N)$-local. For sufficiently large N, it follows from Lemma 3.10 for $\varepsilon=\varepsilon'$ that with probability $1-\delta$ over the draw of G from $G(m,n,\mathcal{D},\Delta)$ and the randomness of h, both

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\mathcal{V}(G(\pi_i)) \geq (1-\varepsilon')\mathop{\mathbb{E}}_{\pi\sim\Pi_k}[\mathcal{V}(G(\pi))] \geq (1-\varepsilon)\cdot\mathcal{V}(G)$$

and $h(G) = \frac{1}{\ell}\sum_{i=1}^{\ell}\mathcal{V}(G(\pi_i))$. □

B. Experimental Details

B.1. Value-to-go computation

We provide pseudo-code for computing value-to-go:

Algorithm 1 $\mathcal{V}(S,t)$

```
Input: Unmatched offline node set $S$, timestep $t$, map $M$ for memoizing intermediate computation, probability vector $\boldsymbol{p}$
if $|S| = 0$ or $t = m+1$ then
   return 0
end if

if $(S,\ t+1)\notin M$ then
   $M[(S,\ t+1)] = \mathcal{V}(S,t+1)$
   for $u\in\mathcal{N}(t)\cap S$ do
      if $(S\setminus\{u\},\ t+1)\notin M$ then
         $M[(S\setminus\{u\},\ t+1)] = \mathcal{V}(S\setminus\{u\},t+1)$
      end if
   end for
end if

$v_{max} = \max_{u\in N(t)\cap S} M[(S\setminus\{u\},\ t+1)]$
return $(1-\boldsymbol{p}_t)\cdot M[(S,\ t+1)] + \boldsymbol{p}_t\cdot\max\{M[(S,\ t+1)], v_{max}\}$
```

B.2. Graph generation

B.2.1. Random graph families

Erdős-Rényi (ER) [Erdős and Rényi, 1960]. Given parameters (m, n, p), we generate a bipartite graph G on m online and n offline nodes where the edge between each (online, offline) node pair appears independently with probability p. Edge weights are sampled from the uniform distribution $U(0,1)$.

Barabási-Albert (BA) [Albert and Barabási, 2002]. We use a process similar to the one described in [Borodin et al., 2020] to generate scale-free bipartite graphs. Given parameters (m, n, b), we generate a bipartite graph G on m online and n offline nodes via a preferential attachment scheme:

1. Start with all n offline nodes.

2. For each online node, attach it to b offline nodes sampled without replacement, where the probability of selecting offline node u is proportional to

$$\Pr[u] = \frac{\text{degree}(u)}{\sum_{u'} \text{degree}(u')}.$$

Similarly, to ER, edge weights are sampled from the uniform distribution $U(0, 1)$.

Geometric (b-RGG). Given parameters (m, n, q) with $q \in [0, 1]$, we generate a bipartite graph G on m online and n by doing the following:

1. Assign each online and offline node u to a uniform random position p_u in $[0, 1]^2$.

2. Connect online node v to offline node w such that

$$w_{vw} \propto -\|p_v - p_w\|_2.$$

3. Only keep the q fraction of edges with the largest weight.

B.2.2. Semi-synthetic and real-word graphs

OSMnx rideshare (Rideshare). We generate a semi-synthetic ridesharing dataset using the OSMnx library [Boeing, 2017]. This dataset generation process is very similar to the one for b-RGG. To make it closer to a real-world application, we replace distances between random points with the time to drive between intersections in a city.

For a given city and parameters (m, n, t), we uniformly sample intersections from a street map layout to generate locations for n drivers and m riders. There is an edge between driver i and rider j if the drive time from i to j is below some threshold t (in practice, t is set to 15 minutes). Approximate drive times are computed using the OSMnx library. Finally, edge weights w_{ij} are generated such that

$$w_{ij} \propto -(\text{drive time from } i \text{ to } j).$$

This dataset can be thought of as a simple ridesharing application in a city. Drivers are idling, waiting to be matched to riders who arrive online at known locations. The application's goal is to minimize the sum travel time between all driver-rider pairs or, equivalently, to maximize $\sum_{e \in M} w_e$ where M is the online matching created by the algorithm. The threshold t is set to avoid riders having to wait too long for a car.

In practice, we use cities of varying sizes, from several thousands of inhabitants (e.g. Piedmont, California) to several hundreds of thousands of inhabitants (e.g. Fremont, California).

gMission. gMission is a spatial crowdsourcing dataset where offline workers are matched to tasks that arrive online. There is an edge between a worker u and a task v if the worker can perform that task. The associated weight w_{uv} is the expected payoff the worker will get from that task, computed based on some distance metric between the task's and the worker's feature vectors. We note that this setting is very similar to the Random Geometric Graphs we prove results for. Inputs are random node-induced subgraphs of the gMission base graph, which is made available by Alomrani et al. [2022].

B.3. Node, edge, and graph features

We augment our graphs with several node-level and graph-level features that the GNN can leverage to improve its predictions.

Node features. On a particular instance, the GNN underlying MAGNOLIA makes a decision for each arrival of a new online node. As the current "matching state" evolves over time, some node features remain unchanged while others are dynamic. Static node features include a positional encoder for the nodes, a one-hot encoding for the skip node, and a binary mask for the offline nodes. In this way, the GNN can (1) differentiate each node from all others, (2) recognize the skip node as being different from other offline nodes, and (3) discriminate online from offline nodes. Dynamic node features include a one-hot encoding for the node the GNN is currently matching, and an arrival probability vector that is updated to 1 (respectively, 0) for nodes that have already arrived (respectively, not arrived) in the run of the algorithm.

Edge features. The weight w_{ij} of each edge (i, j) is encoded as a 1-dimensional edge feature.

Graph features. We use a single graph-level feature: the ratio of remaining unmatched online nodes to offline nodes. Intuitively, an algorithm for online bipartite matching should get more greedy as this ratio goes down since greedy decisions are unlikely to lead to later conflicts.

B.4. Architecture

The convolutional layers of our GNN follow a GENConv architecture [Li et al., 2020] and its implementation in PyTorch Geometric [Fey and Lenssen, 2019]. The embedding update rule for this architecture mirrors the functional form of the dynamic program representation of value-to-go:

$$h_v^{(k)} = \text{MLP}\left(h_v^{(k-1)} + \max_{u \in \mathcal{N}(v)} \left\{ \text{ReLU}(h_u^{(k-1)} + w_{vu}) \right\}\right).$$

We compare different GNN architectures for VTG approximation in Appendix B.7.

B.5. Training error and model accuracy

We train our model using mean squared error. On each training sample, the model is given a graph instance and the current online node t. It then tries to predict the value-to-go of all nodes in the graph. The only valid actions on step t are to either match t to one of its neighbors or not to match t which is represented by matching t to the skip node. Hence, the model's prediction is masked to only consider the neighbors of t (which include the skip node) and we compute the mean squared error between those predictions and the actual value-to-go values given by the online optimal algorithm.

Model accuracy is used for hyperparameter tuning and is a good metric for the empirical performance of the GNN when used as an online matching algorithm. It is simply computed as the percentage of times the GNN chooses the same action as the online optimal algorithm. Here, choosing the same action could either mean matching to the same offline node or skipping the online node.

B.6. Hyperparameter tuning

We perform hyperparameter tuning using a validation set of size 300. We perform around 1000 trials, tuning the parameters as described in Table 1. Each trial is evaluated by its validation set accuracy. The hyperparameters are tuned with Bayesian search [Snoek et al., 2012] and pruning from the Optuna library [Akiba et al., 2019] to stop unpromising runs early. Similarly to the training setup, the hyperparameter tuning is done on small graphs (10×6) and (6×10) even though the eventual testing may be on larger graphs. All the training was done on an NVIDIA GeForce GTX Titan X.

B.7. MAGNOLIA using different architectures

One of MAGNOLIA's strengths is that it is a modular pipeline that can accept any GNN architecture as a VTG approximator. In Figure 4, we validate the choice of the GENConv architecture by including a comparison with various state-of-the-art GNN models [Li et al., 2020, Morris et al., 2019, Brody et al., 2022]. Note that GENConv and DeeperGCN have the same underlying GNN but use different layers and aggregation functions. We observe that all models achieve similar competitive ratios, with GENConv and DeeperGCN performing slightly better.

Table 1. Hyperparameter ranges

	Hyperparameter	Values
GNN	$\#$ of message passing layers	$\{1, \ldots, 6\}$
	# of MLP layers	$\{1, \ldots, 5\}$
	Hidden dimension size	$\{2^i \mid i \in \{1, \ldots, 6\}\}$
	Dropout	[0, 0.5]
Training	Batch size	$\{2^i \mid i \in \{1, \ldots, 6\}\}$
	Epochs	$\{2^i \mid i \in \{1, \ldots, 8\}\}$
	Learning rate	$[1e-5, 1e-10]$

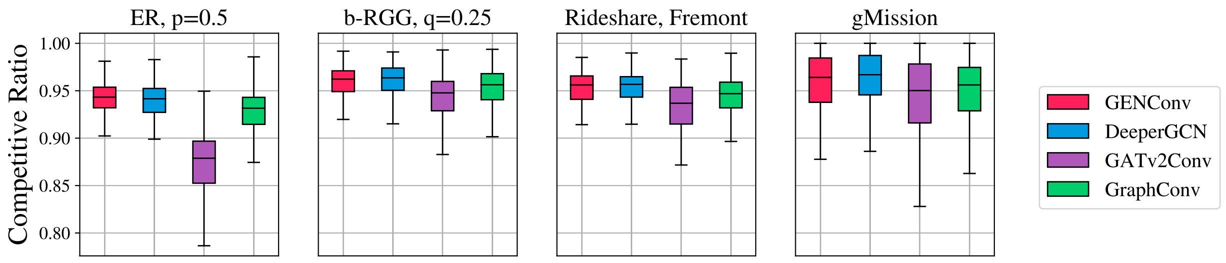

Figure 4. Boxplot showing the distribution of competitive ratios for MAGNOLIA with different underlying GNN architectures across graph configurations. All graphs are of size (10×20).

B.8. Complete results for Section 4.2

For the results in Appendices B.8.1 to B.8.3, the GNN underlying MAGNOLIA is trained on a set of 2000 instances of graphs from ER with $p = 0.75$, BA with $b = 4$, and GEOM with $q = 0.25$ of size (6×10). Reported results are distributions and averages of competitive ratios from 6000 OBBM unseen instances of size (10×30) across 12 graph configurations. In particular, we elected to train on a subset of graph configurations since this considerably improves training time, and in our experience, leads to similar results.

The results in Appendix B.8.4 come from a GNN-based meta model which is given as input two GNNs trained on graphs of size (6×10) and (10×6), respectively. The meta-GNN is trained on the competitive ratios achieved by each GNN on 2000 instances of graphs from ER with $p = 0.75$, BA with $b = 4$, and GEOM with $q = 0.25$, each across graph sizes (10×6), (8×8), and (6×10). Whereas the GNN-based model selects a GNN to run each instance on using predicted competitive ratios, the threshold-based meta algorithm simply runs on one GNN if the ratio of online nodes to offline nodes exceeds a fixed threshold t. Empirically, we found that $t = 1.5$ performs well. Evaluation for both models once again happens over 6000 instances from the 12 graph configurations.

Finally, in Appendix B.8.5, we train MAGNOLIA with a different GNN on each possible noise level ρ. The training and evaluation specifications for each of these noise-dependent GNNs are the same as those from Appendix B.8.1.

B.8.1. Magnolia makes good decisions

Table 2. Average competitive ratio by graph configuration with node ratio (10×20).

	Parameter	GNN	Greedy	Threshold Greedy	LP
ER	$p = 0.25$	**0.945**	0.881	0.887	0.929
	$p = 0.5$	**0.943**	0.883	0.897	0.917
	$p = 0.75$	**0.949**	0.905	0.914	0.915
BA	$b = 4$	**0.937**	0.857	0.875	0.921
	$b = 6$	**0.944**	0.885	0.896	0.916
	$b = 8$	**0.955**	0.911	0.922	0.921
GEOM	$q = 0.15$	**0.978**	0.938	0.938	0.958
	$q = 0.25$	**0.961**	0.922	0.922	0.939
	$q = 0.5$	**0.950**	0.924	0.924	0.921
RIDESHARE	city = Piedmont	**0.957**	0.935	0.939	0.936
	city = Fremont	**0.957**	0.929	0.933	0.930
GMISSION	-	**0.951**	0.929	0.802	**0.951**

B.8.2. MAGNOLIA SHOWS SIZE GENERALIZATION

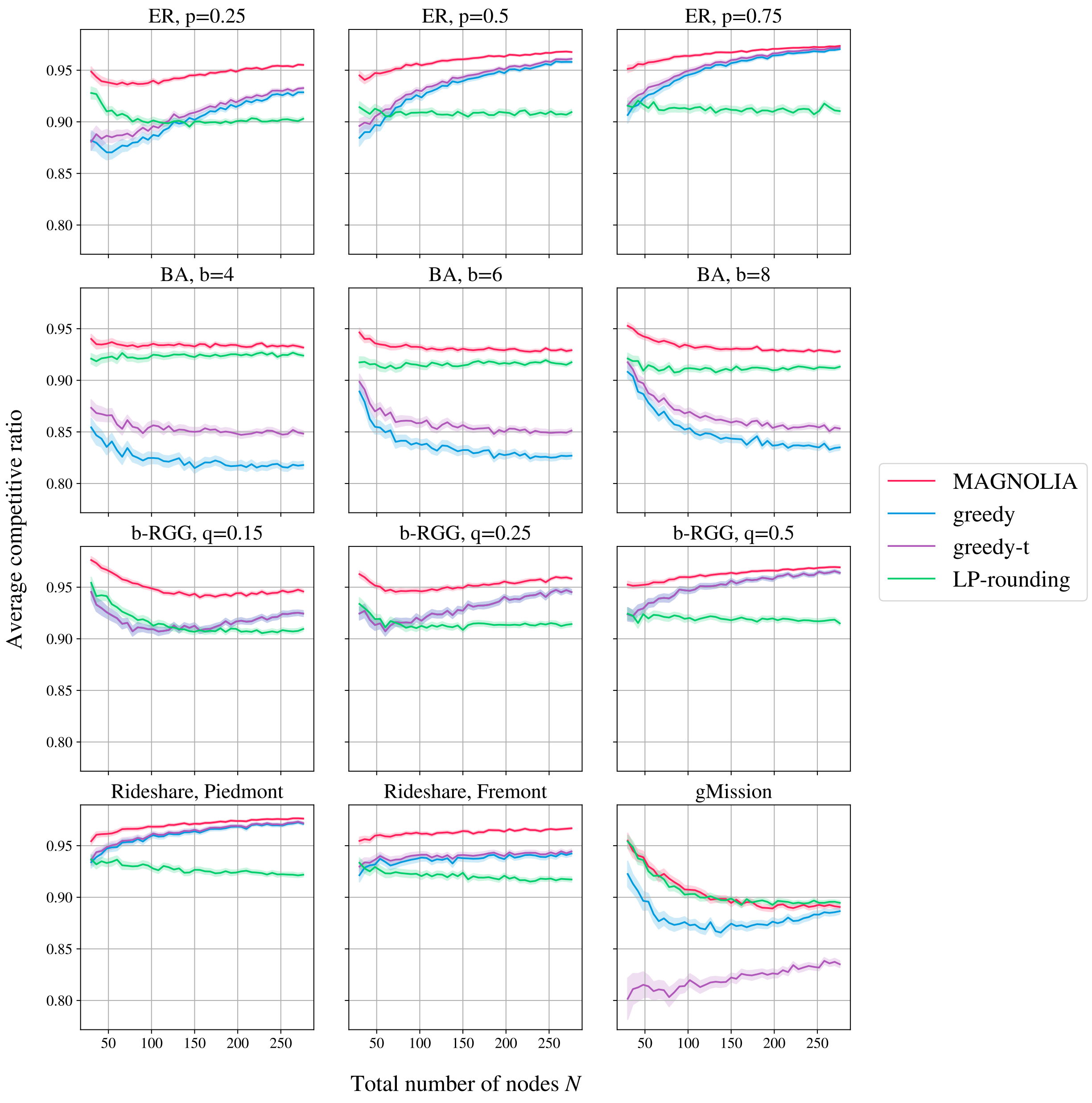

Figure 5. Evolution of competitive ratio over graphs of increasing size for a GNN trained on graphs of size (6×10). All test graphs have a 2:1 ratio of online to offline nodes.

B.8.3. REQUIREMENTS FOR SIZE GENERALIZATION

One of the advantages of our approach is that it performs well when trained on small graphs. Indeed, Appendix B.8.2 shows that MAGNOLIA exhibits size generalization. Two questions remain:

1. Would we observe better performance if MAGNOLIA was trained on larger graphs?
2. How small can the training graphs be while still exhibiting size generalization?

To address these questions, we compare the size generalization of MAGNOLIA when trained on graphs of varying size. We see in Figure 6 that MAGNOLIA shows strong generalization to graph size, even when trained on very small graphs. We observe that, surprisingly, the GNN trained on the smallest graphs (5×3) performs the best on gMission. This can be explained by the fact that (5×3) graphs are more likely to be sparse, making them similar to the very sparse gMission inputs.

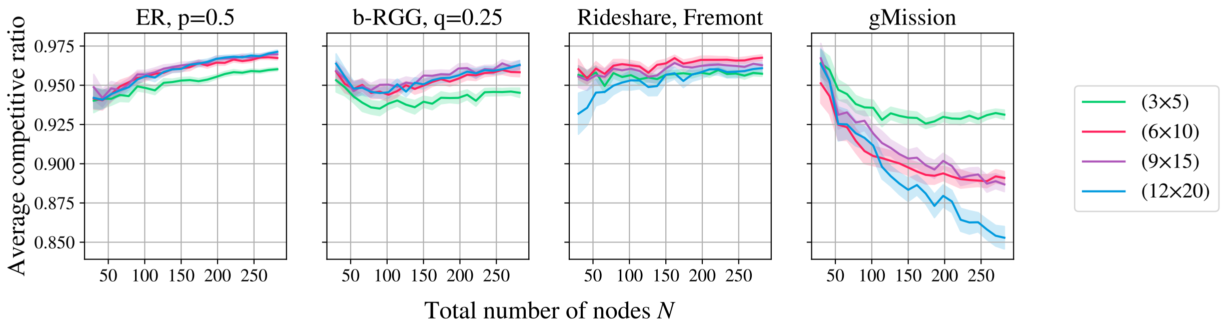

Figure 6. Evolution of competitive ratio over graphs of increasing size for GNNs trained on graphs of different sizes with the same (6:10) ratio. All test graphs have a 1:2 ratio of offline to online nodes.

B.8.4. Meta-model improves regime generalization

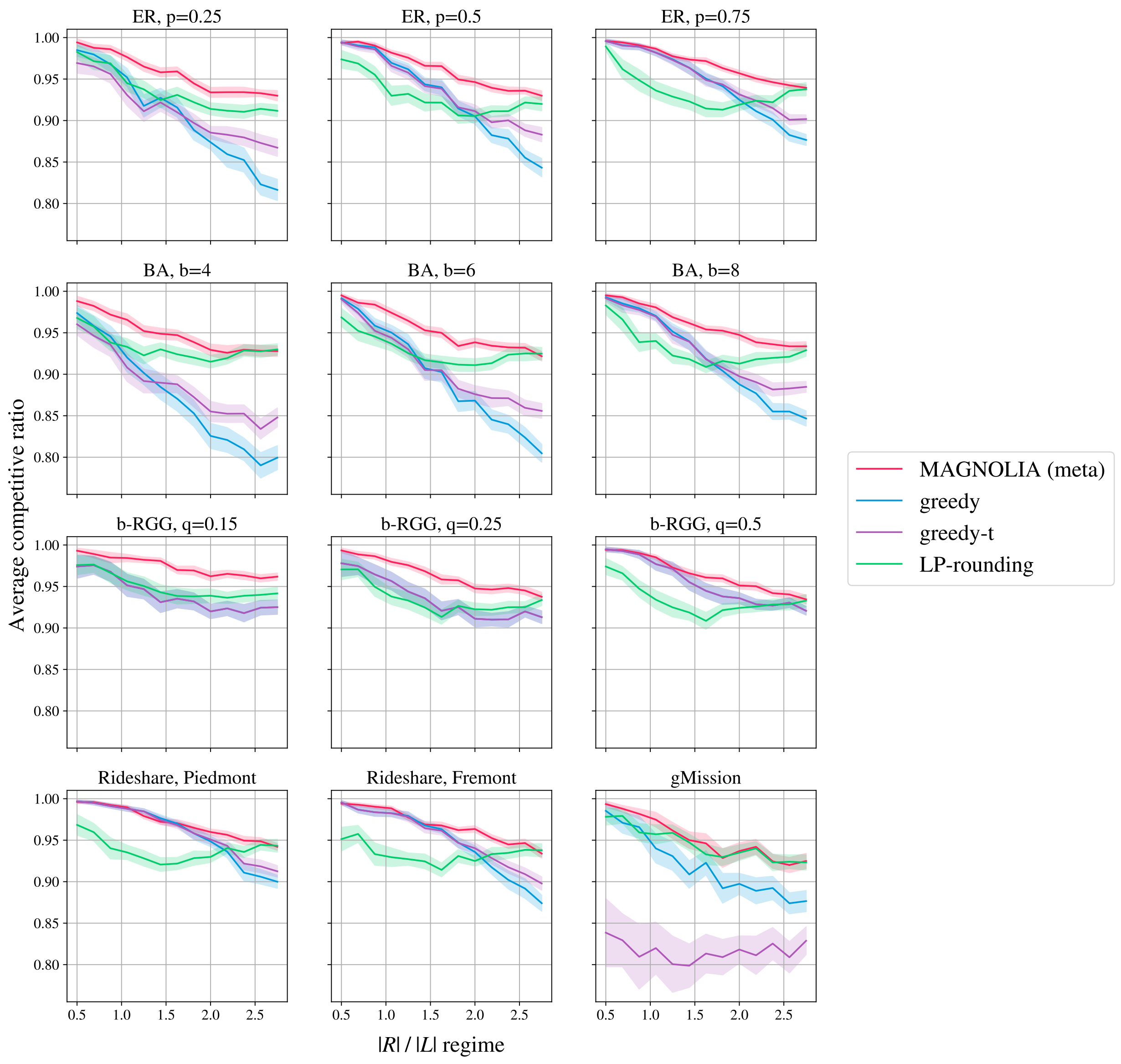

Figure 7. Evolution of competitive ratio over regimes for MAGNOLIA enabled with a meta-GNN. For evaluation, $|L|$ is kept fixed at 16 offline nodes, and $|R|$ varies from 8 to 64 online nodes

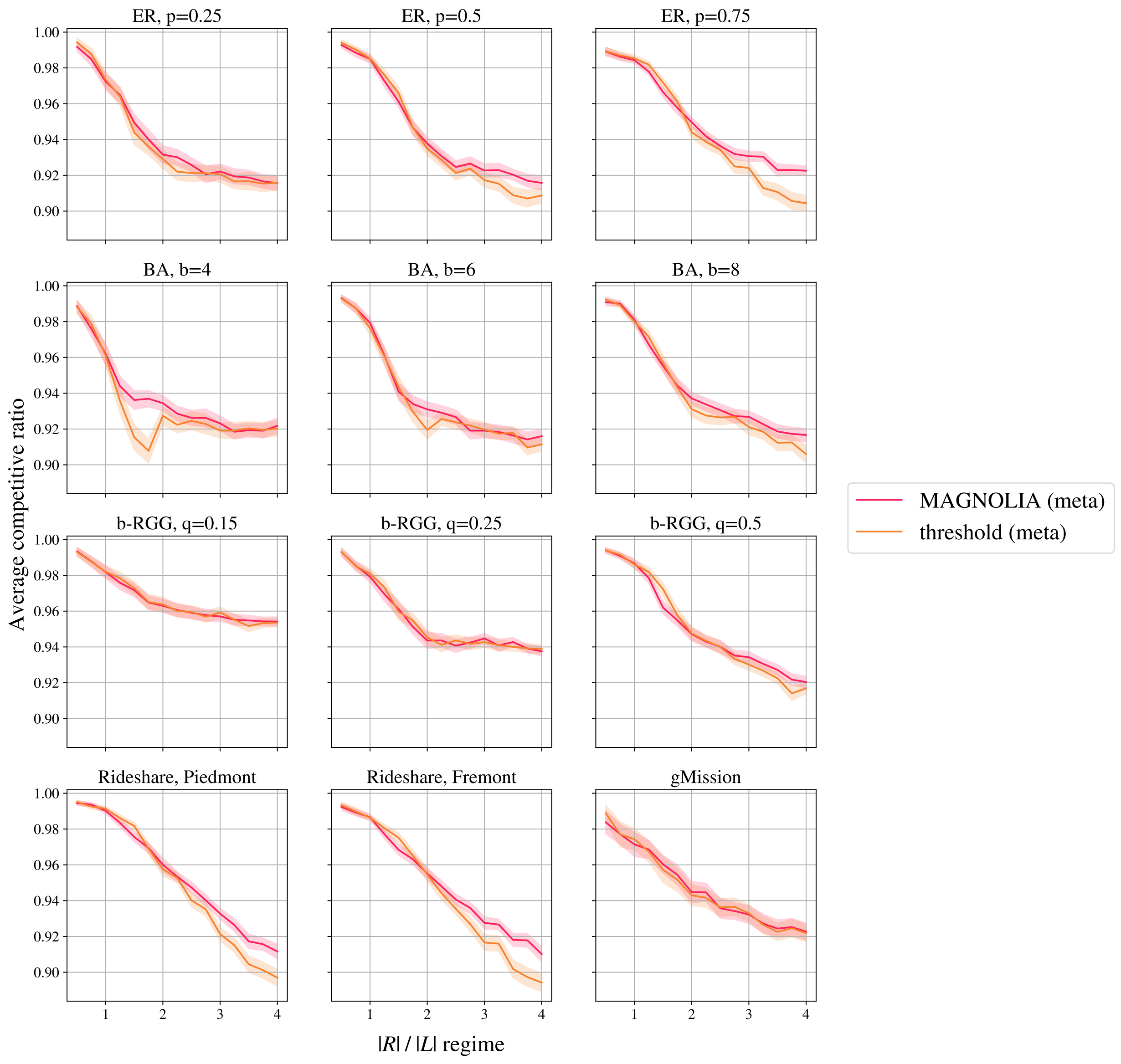

Figure 8. Evolution of competitive ratio over regimes for MAGNOLIA enabled with a meta-GNN against simple threshold model. For evaluation, $|L|$ is kept fixed at 16 offline nodes, and $|R|$ varies from 8 to 64 online nodes

B.8.5. MAGNOLIA IS ROBUST TO NOISY INPUTS

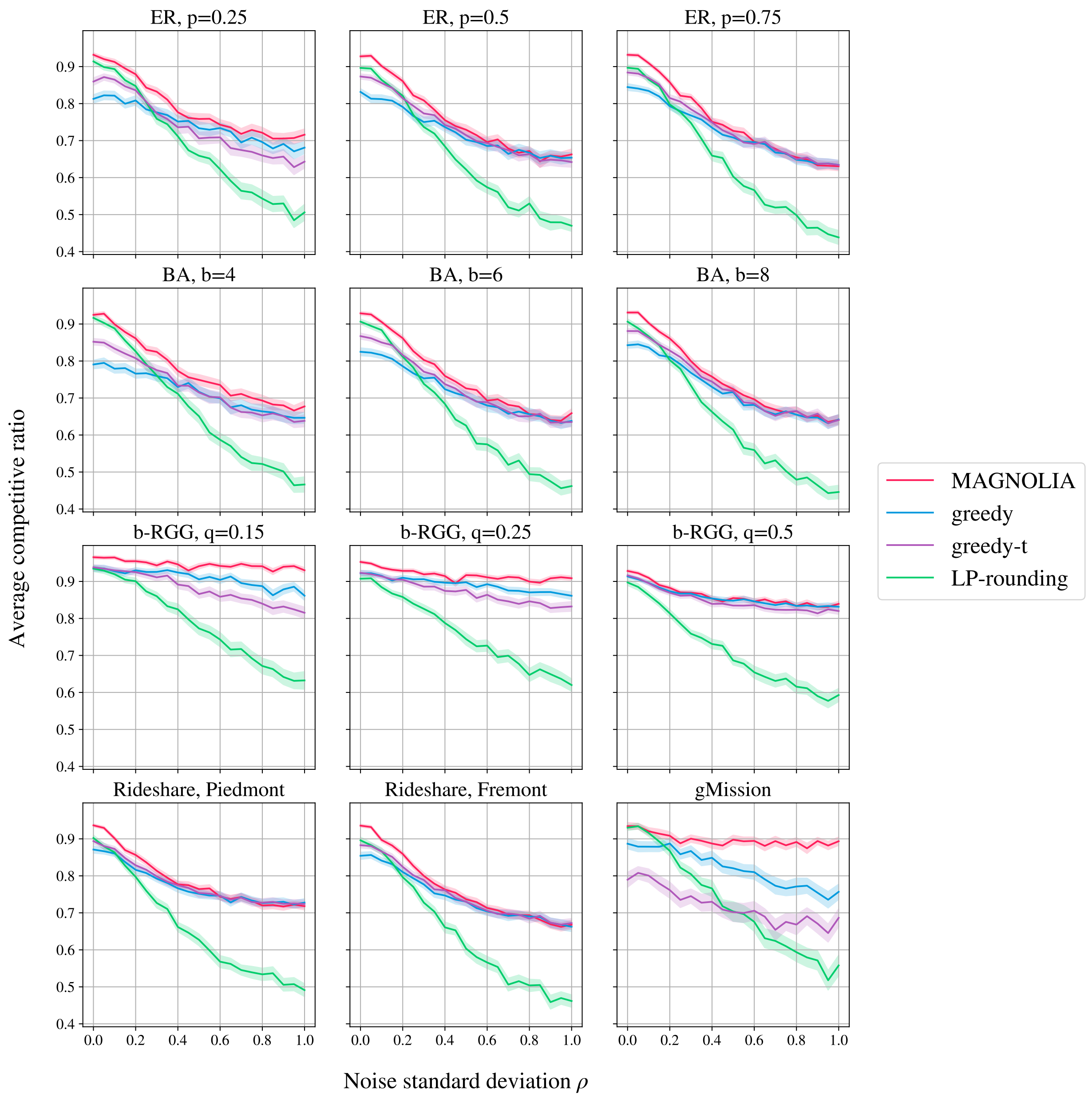

Figure 9. Evolution of competitive ratio as a function of noise level ρ for graphs of size (10×30). A $\mathcal{N}(0, \rho^2)$ noise is added independently to each edge weight and arrival probability.