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

Alexandre Hayderi [1] **Amin Saberi** [2] **Ellen Vitercik** [1 2] **Anders Wikum** [2]

Abstract

Online Bayesian bipartite matching is a central problem in digital marketplaces and exchanges, including advertising, crowdsourcing, rideshare, and kidney exchange. We introduce a graph neural network (GNN) approach that emulates the problem's combinatorially-complex 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 final matching if the algorithm takes that action, then acts optimally in the future. We train a GNN to estimate VTG and show empirically that this GNN 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 within local neighborhoods in the graphs. This structure matches the local behavior of GNNs, providing theoretical justification for our approach.

1. Introduction

Online matching is a critical 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.

[1]Department of Computer Science, Stanford University, Stanford, CA, USA [2]Department of Management Science & Engineering, Stanford University, Stanford, CA, USA. Correspondence to: Anders Wikum <wikum@stanford.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

In the notoriously challenging *Online Bayesian Bipartite Matching (OBBM)* problem, there is a bipartite graph with *online* and *offline* nodes and weighted edges. The node sets could, for example, represent crowdworkers and tasks, with weights encoding workers' payoffs for completing tasks. Matching occurs over a series of rounds, with one online node arriving with known probability in each round. Upon a successful arrival, a matching algorithm must determine whether to match the node to an offline node. Alternatively, the algorithm can skip the node, in which case it will never be matched, leaving nodes available for the future. The goal is to compute a high-weight matching. However, it is not known which online nodes will arrive *a priori*.

The online optimal algorithm for OBBM, OPT_{on}, knows the distribution over online node arrivals but not the future realization of arriving nodes. 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 the future node arrivals. In more detail, OPT_{on} can be formulated as a dynamic programming (DP) routine. At each timestep, it computes the *value-to-go (VTG)* of each action, which is the expected final matching weight if it (1) takes that action and then (2) takes each subsequent action to maximize the expected weight of the final matching. The DP routine takes the action with the highest VTG.

1.1. Our contributions

We train a graph neural network (GNN) that empirically competes with this omnipotent optimal algorithm by estimating the VTG of each action. Moreover, we provide a theoretical analysis that helps justify this deep learning architecture's suitability for the OBBM problem.

Key challenges. The primary obstacle we face is the sheer complexity of OBBM: it takes exponential space in the worst case to provide even a constant-factor approximation to OPT_{on} (Papadimitriou et al., 2021). Nonetheless, we show empirically that GNNs compete with OPT_{on} across many tasks. Moreover, although deep learning architectures are notoriously difficult to analyze theoretically, we prove a correspondence between the functions computable by GNNs

and the VTG function. We prove that for a broad family of graph distributions, VTG can be efficiently approximated by aggregating information across only small local neighborhoods in the graphs. This structure matches the behavior of GNNs: over a series of rounds, each node computes a *message*, which is a function of its *internal representation* (a feature vector), passes this message to its neighbors, and updates its internal representation using its received messages. Thus, GNNs are ideal for approximating functions that only depend on a graph's local neighborhood.

We summarize our two primary contributions as follows.

Theoretical guarantees. We prove that for *bipartite random geometric graphs (b-RGGs)*, VTG can be efficiently approximated by only aggregating information within small local neighborhoods. In a b-RGG, edges are formed based on the similarity of node embeddings within a latent space. Thus, b-RGGs often arise in spatial crowdsourcing (Tong et al., 2020) such as ridesharing, which necessitates physical proximity between drivers and riders. We prove that b-RGGs can be decomposed into smaller, local subgraphs with limited interconnectivity. This structure allows us to prove that VTG can be estimated by a function that only aggregates information within these local subgraphs.

Empirical analysis. We present MAGNOLIA (**M**atching **A**lgorithms via **GN**Ns for **Onli**ne Value-to-go **A**pproximation), a GNN-based online matching framework that mimics the actions of OPT_{on} by predicting the VTG of each feasible matching decision. While computing VTG is intractable for even moderately sized graphs, we show empirically that a supervised learning approach is still effective; despite being trained on graphs with 16 total nodes, MAGNOLIA beats state-of-the-art baseline algorithms across a broad range of problem types, sizes, and regimes, showing strong out-of-distribution generalization.

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 matching in hindsight. However, better approximation ratios are possible when competing with the optimal *online* algorithm OPT_{on}. Papadimitriou et al. (2021) gave a 0.51-approximation algorithm for OPT_{on}, a bound subsequently improved by Braverman et al. (2022) and Naor et al. (2023). Our experiments demonstrate that our approach consistently outperforms these approximation algorithms.

ML for combinatorial optimization. There has been significant recent interest in integrating ML with combinatorial optimization (see, e.g., the surveys by Bengio et al. (2021) and Cappart et al. (2023)). 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). The work by Alomrani et al. (2022) is one of the few that studies average-case performance. They present an end-to-end RL framework for learning online bipartite matching policies in the unknown i.i.d. arrival setting using GNNs. Their approach differs from ours in a few critical ways. (i) Whereas the structure of the optimal online algorithm is not known in the unknown i.i.d. setting, in OBBM, the optimal online algorithm is simple to express but computationally intractable, resulting in a fundamentally different ML task. (ii) The existence of good approximation algorithms for OBBM allows us to compare MAGNOLIA's performance to stronger benchmarks than those available in the unknown i.i.d. setting. (iii) Alomrani et al.'s paper is empirical, analyzing the performance of various models to identify which underlying characteristics make them perform well. In contrast, we provide theoretical justification for a GNN's ability to replicate the decisions of an optimal online algorithm on real-world graphs, as well as experiments. Li et al. (2023) bridge the gap between worst-case guarantees and average-case performance for online matching by switching between expert and ML predictions online. Using VTG predictions from MAGNOLIA within this switching framework yields an algorithm which is competitive against any fixed online algorithm.

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 $E \subseteq L \times R$. When the underlying graph G is not clear from context, we refer to these sets as $L(G)$, $R(G)$, and $E(G)$. We use the notation $\mathcal{N}_G(t)$ to denote the neighbors of online node $t \in R$ in G, and $N = m + n$ to denote the total number of nodes.

2.1. Online Bayesian bipartite matching (OBBM)

An input to the OBBM problem is a bipartite graph $G = (L, R, E)$ attributed with edge weights $\{w_{ij}\}_{(i,j)\in E}$ and online node arrival probabilities $\{p_i\}_{i\in R}$. 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 irrevocably decide to match t with an unmatched offline neighbor or skip t and not match it to any node. The goal is to maximize the total weight of the final matching.

Although an algorithm for OBBM knows the input graph G from the onset, it 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} makes decisions that maximize the expected weight of the matching it returns over the randomness of the online node arrivals. In detail, OPT_{on} computes the Bellman equation—or *value-to-go function*—$\mathcal{V}_G(S,t)$, which is the expected value of the maximum weight matching achievable on G over sequential arrivals $\{t,\dots,m\}$, with matchings restricted to the set of remaining offline nodes S. Additional matches are only possible when there are available offline and online nodes, so $\mathcal{V}_G(\emptyset,t)=0$ for all $t\in R$ and $\mathcal{V}_G(S,m+1)=0$ for all $S\subseteq L$. 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 $\mathcal{V}_G(S,t,u)=w_{tu}+\mathcal{V}_G(S\backslash\{u\},t+1)$ is the utility of matching t to u: the edge (t,u) is added to the matching and u is subsequently no longer available. Conditioned on node t not arriving, the maximum expected matching value achievable over online nodes $\{t,\dots,m\}$ and offline node set S is $\mathcal{V}_G(S,t+1)$. Conditioned on node t arriving, OPT_{on} either matches t with the offline node u that maximizes utility or skips t, depending whether $\max_{u\in\mathcal{N}_G(t)\cap S}\mathcal{V}_G(S,t,u)$ or $\mathcal{V}_G(S,t+1)$ is larger. We include pseudo-code for computing $\mathcal{V}_G(S,t)$ in Algorithm 1.

Throughout the paper, $\mathcal{V}(G):=\mathcal{V}_G(L(G),1)$ refers to the full value-to-go computation on the input graph G, i.e., the expected value of the matching returned by OPT_{on} when run on G. If H is the node-induced subgraph of G over the vertex set $S\cup\{t,\dots,|R|\}$, then $\mathcal{V}(H)=\mathcal{V}_G(S,t)$. This observation will help to simplify some of the analysis; rather than proving statements over all $S\subseteq L$ and $t\in R$ for some larger graph G, we will instead prove statements about $\mathcal{V}(H)$ over all attributed graphs H.

2.2. Graph neural networks

Let $G=(V,E,\mathcal{X})$ be a graph with node attributes $\mathcal{X}\subseteq\mathbb{R}^d$, so each node $v\in V$ is associated with an initial embedding $\boldsymbol{h}_v^{(0)}\in\mathcal{X}$. A *Message-Passing Graph Neural Network* (MPNN or GNN) of depth k iteratively computes a sequence of embeddings $\boldsymbol{h}_v^{(1)},\dots,\boldsymbol{h}_v^{(k)}$ for each node $v\in V$ starting from $\boldsymbol{h}_v^{(0)}$. 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 $\boldsymbol{h}_v^{(i)}$ of node v is computed by aggregating the messages $\boldsymbol{m}_u^{(i)}$ from each of v's neighbors:

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

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

These functions can contain learnable parameters, and different choices of these functions lead to different named GNN architectures. This framework also extends to graphs with edge attributes. By design, a single GNN can be trained and evaluated on graphs with any number of nodes.

A key observation is that the embedding $\boldsymbol{h}_v^{(k)}$ of a node v is only a function of the embeddings within a k-hop neighborhood of v. In this way, GNNs are well-suited to learn functions that depend only on local substructures.

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. b-RGGs are a random graph family that mimics the structure of real-world networks by generating edges according to the similarity between node embeddings in some latent space. As such, they are often used in *spatial crowdsourcing* applications such as rideshare, where riders must be matched to nearby drivers (Tong et al., 2020).

In Section 3.1, we prove that b-RGGs can partitioned into small subgraphs with few edges crossing between subgraphs. Next, in Section 3.2, we show that this property implies that VTG is locally approximable. This result aligns with the inherent local processing capabilities of GNNs, offering a theoretical foundation for our methodology. The full proofs of all results are in Appendix A.

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.

These properties are made formal in Theorem 3.7. Achieving both is nontrivial: a fine-grained partition may lead to small subgraphs, but it will likely result in many edges crossing between subgraphs.

Bipartite random geometric graphs. The defining characteristic of b-RGGs is that online and offline nodes are connected only when their embeddings are sufficiently close according to some metric. We prove results for ℓ_∞, though they immediately 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 $\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_\infty \le \Delta$.

A partition π of $[0,1]^d$ induces a partition of b-RGGs into subgraphs: for $G \in \text{supp}(G(m,n,\mathcal{D},\Delta))$, let $G(\pi)$ be the graph obtained from G by removing all edges (i,j) with embeddings $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ in different cells of π. We can thus 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. This shift is taken modulo 1, so the final cells are of equal volume but not necessarily contiguous.

Definition 3.2. *For $k \in \mathbb{Z}_{\ge 1}$, 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 $\boldsymbol{s}_i \in [0, \frac{1}{k}]$.*

We use the notation Π_k to denote the uniform distribution over $(k,\boldsymbol{s})$-partitions with $\boldsymbol{s} \sim \text{Unif}(0, 1/k)^d$ and refer to $\pi \sim \Pi_k$ as a *random k-partition*.

b-RGG decomposition. For carefully chosen k, nearby vectors are likely to lie in the same cell of a random k-partition π. Since the edges of any b-RGG G are based on proximity, this means each edge of G is unlikely to be removed when forming $G(\pi)$.

Lemma 3.3. *Let $\boldsymbol{x_1}, \dots, \boldsymbol{x_N} \in [0,1]^d$, $\varepsilon > 0$, $\Delta \le \frac{\varepsilon}{2d}$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$. If $\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_\infty \le \Delta$, 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 sketch. Notice that $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ lie in different cells of $\pi \sim \Pi_k$ precisely when, in at least one dimension ℓ, some point of $\boldsymbol{s}_\ell + \{0, \frac{1}{k}, \dots, \frac{k-1}{k}\}$ lies in the interval $[(\boldsymbol{x_i})_\ell, (\boldsymbol{x_j})_\ell]$. By symmetry, this occurs in dimension ℓ independently with probability equal to the length of the interval $[(\boldsymbol{x_i})_\ell, (\boldsymbol{x_j})_\ell] \le \Delta$ over the measure $1/k$ of possible shifts $\boldsymbol{s}_\ell$. For $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$, the probability that this occurs in any dimension is at most $1 - (1 - k\Delta)^d \le \varepsilon$. □

Under a mild anti-concentration assumption on $\mathcal{D}$, the number of b-RGG latent embeddings in any cell of a random k-partition with $\Omega(N)$ cells is likely sublinear in N. This does not hold, for example, when $\mathcal{D}$ is a point mass. We avoid pathological cases with the concept of a β-smooth distribution (Haghtalab et al., 2022) from smoothed analysis.

Definition 3.4. *A distribution $\mathcal{D}$ over $[0,1]^d$ with probability density function f is* β-smooth *if $\sup f(\boldsymbol{x}) \le \beta$.*

Our argument is based on a connection to balls-into-bins processes. A classic result guarantees that if N balls are dropped uniformly at random into N bins, the maximum load is $O(\ln N)$ with high probability (Mitzenmacher and Upfal, 2005). Lemma 3.5 slightly modifies this result.

Lemma 3.5. *For $\beta \ge 1$, 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.*

We 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 3.6. *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.*

We now state this section's main theorem.

Theorem 3.7. *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 sketch. (1) follows from Lemma 3.3 and (2) follows from Corollary 3.6. □

3.2. Local approximation of value-to-go

This section shows that the local decomposability of b-RGGs from Theorem 3.7 means VTG can be approximated using *local graph functions*. At a high level, these are functions that can be computed using only information from small neighborhoods.

Definition 3.8 (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 3.9. *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 $|f(G) - h(G)| \le \varepsilon f(G)$ with probability $1 - \delta$ over $G \sim \mathcal{G}$ and any randomness in h.*

An initial (non-local) approximation. First, we prove that the VTG of an OBBM instance can only decrease after removing the edges needed to form $G(\pi)$.

Lemma 3.10. *For any $G \in \text{supp}(G(m, n, \mathcal{D}, \Delta)$ and any hypercube partition π, $\mathcal{V}(G(\pi)) \leq \mathcal{V}(G)$.*

Proof sketch. We show this via induction on the number of online nodes. When G has no online nodes, $\mathcal{V}(G) = \mathcal{V}(G(\pi)) = 0$. The inductive step uses the one-step DP representations of $\mathcal{V}(G) = \mathcal{V}_G(L, 1)$ and $\mathcal{V}(G(\pi)) = \mathcal{V}_{G(\pi)}(L, 1)$. In particular, if G has t online nodes, then $\mathcal{V}_G(L, 2)$ and $\mathcal{V}_G(L \setminus \{u\}, 2)$ are full VTG computations on subgraphs of G with $t-1$ online nodes. Applying the inductive hypothesis gives that $\mathcal{V}_G(L, 2) \geq \mathcal{V}_{G(\pi)}(L, 2)$ and $\mathcal{V}_G(L, 2) \geq \mathcal{V}_{G(\pi)}(L \setminus \{u\}, 2)$. Along with the fact that $\mathcal{N}_{G(\pi)}(1) \subseteq \mathcal{N}_G(1)$ as $G(\pi)$ is formed from G by removing edges, these bounds imply $\mathcal{V}(G) \geq \mathcal{V}(G(\pi))$. □

To establish a lower approximation bound, given $G \in \text{supp}(G(m, n, \mathcal{D}, \Delta))$, we show that because each edge in 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 3.11. *For any $G \in \text{supp}(G(m, n, \mathcal{D}, \Delta))$, $\varepsilon > 0$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$, $\mathbb{E}_{\pi \sim \Pi_k}\left[\mathcal{V}(G(\pi))\right] \geq (1-\varepsilon)\mathcal{V}(G)$.*

Proof sketch. First we show that $\mathcal{V}(G)$ decomposes across edges. Recall that $\mathcal{V}(G)$ is the expected value of $\text{OPT}_{on}(G)$. Given an arrival sequence $\boldsymbol{a} \in \{0,1\}^m$, OPT_{on} outputs a deterministic matching $M(\boldsymbol{a})$. Therefore,

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

After rearranging, we find $\mathcal{V}(G) = \sum_{e \in E} \alpha_e w_e$ for

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

Crucially, for any hypercube partition π,

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

The right-hand side is the expected value of the matching returned by an online algorithm on $G(\pi)$ which, for any $\boldsymbol{a}$, outputs $M(\boldsymbol{a}) \cap E(G(\pi))$. The left-hand side is the expected value of OPT_{on} $(G(\pi))$. The lemma statement then follows from Equation (1) and Theorem 3.7. □

Lemmas 3.10 and 3.11 show that the function $\tilde{\mathcal{V}}(G) = \mathbb{E}_{\pi \sim \Pi_k}\left[\mathcal{V}(G(\pi))\right]$ can approximate $\mathcal{V}(G)$ with high accuracy over any b-RGG, but $\tilde{\mathcal{V}}(\cdot)$ may not be r-local for any reasonable value of r. The main obstacle to overcome is that this would require the connected components of $G(\pi)$ to be of size at most r under all partitions $\pi \in \text{supp}(\Pi_k)$.

Local approximation via Monte Carlo estimation. The key observation is that unlike $\tilde{\mathcal{V}}(\cdot)$, a random function that samples ℓ partitions $\pi_1, \ldots, \pi_\ell \sim \Pi_k$ and outputs the sample estimate $\frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{V}(G(\pi_i))$ achieves r-locality by only requiring that the connected components of $G(\pi_1), \ldots, G(\pi_\ell)$ are of size at most r. We show that it is possible to choose ℓ such that with high probability over the joint draw of G and the draw of ℓ partitions, the sample estimate is sufficiently accurate and still a local function.

Lemma 3.12. *Let $\mathcal{D}$ be a β-smooth distribution and $\Delta = O(N^{-1/d})$ where N is sufficiently large. For all $\varepsilon \in (0, 1/2]$, 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 the draw of $\ell \geq \frac{2}{\varepsilon^2} \log \frac{4}{\delta}$ 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 sketch. To simplify notation, let $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_\ell\}$, $\mathcal{G} = G(m, n, \mathcal{D}, \Delta)$, $S_\ell(G, \boldsymbol{\pi}) = \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{V}(G(\pi_i))$, and $\bar{S}(G) = \mathbb{E}_\pi[\mathcal{V}(G(\pi))]$. For $G \in \text{supp}(\mathcal{G})$ and $\boldsymbol{\pi} \in \text{supp}(\Pi_k)$, let $A(G, \boldsymbol{\pi})$ be the event that $S_\ell(G, \boldsymbol{\pi}) \geq (1 - \varepsilon)\bar{S}(G)$. Moreover, let $B(G, \boldsymbol{\pi})$ be 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.

By the tower property, we can rewrite

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

By Theorem 3.7 and a union bound over the ℓ partitions,

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

for N sufficiently large.

Next, for fixed $G \in \text{supp}(\mathcal{G})$ 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 3.10. A Hoeffding bound then implies that $\Pr_{\boldsymbol{\pi} \sim \Pi_k}[A(G, \boldsymbol{\pi})^{\complement} \mid G] \leq \delta/2$. This bound and Equation (2) imply the lemma statement. □

Finally, we give our main theorem.

Theorem 3.13. *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 $\delta \in (0, 1]$.*

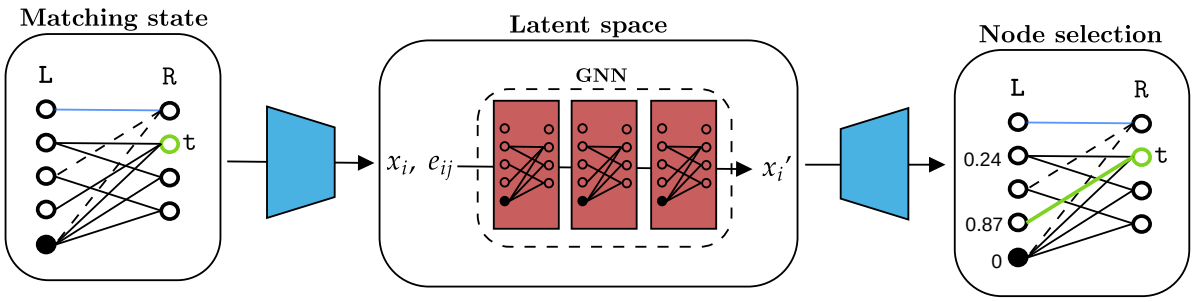

Figure 1. MAGNOLIA's GNN-based matching subroutine.

Proof sketch. Let $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$ and $\varepsilon' = 1 - \sqrt{1-\varepsilon}$ so $(1-\varepsilon')^2 = 1 - \varepsilon$. By Lemma 3.11 with $\varepsilon = \varepsilon'$, $\mathbb{E}_{\pi \sim \Pi_k}[\mathcal{V}(G(\pi))] \geq (1-\varepsilon')\mathcal{V}(G)$ for all $G \in \text{supp}(G(m, n, \mathcal{D}, \Delta))$. By Lemma 3.12, $\mathbb{E}[\mathcal{V}(G(\pi))]$ can be approximated to error $1 - \varepsilon'$ via a local Monte Carlo estimate. This implies the theorem statement. □

4. Experiments

In this section, we introduce MAGNOLIA[1] and demonstrate its performance against state-of-the-art baselines across a broad range of graph families and problem regimes. We describe our model and experimental setup in Sections 4.1 and 4.2, then present results from experiments in Section 4.3. More implementation details can be found in Appendix B.1.

4.1. Model

Learned matching model. Recall that for an OBBM input G, arriving online node t, and set of offline nodes S, the online optimal algorithm OPT_{on} skips t if $\mathcal{V}_G(S, t+1) > \max\{\mathcal{V}_G(S, t, u) : u \in \mathcal{N}_G(t) \cap S\}$ and otherwise matches t to the neighbor that maximizes $\mathcal{V}_G(S, t, u)$. Fundamentally, OPT_{on} is a greedy algorithm with respect to VTG. MAGNOLIA replaces the VTG computations in OPT_{on} with approximations from a GNN. See Figure 1 for an overview. To start, the current *matching state* is encoded as an attributed graph. A matching state includes the input graph G, arriving online node t, and set of available offline nodes S. This attributed graph is fed into a GNN, which outputs an approximate VTG associated with each feasible action. Finally, the decision with the highest predicted VTG is chosen. This process is repeated on all matching states encountered while running on G until no online nodes remain.

Training protocol. The GNN underlying MAGNOLIA is trained to approximate VTG via supervised learning. Since computing targets involves solving an exponential-sized DP, we construct a training set of 2000 instances on 16 nodes. We generate matching states and targets from each instance via teacher forcing by following the decisions of OPT_{on}.

[1] https://github.com/anders-wikum/GNN-OBM

4.2. Experimental setup

Instance generation. Training and testing are done on synthetic and semi-synthetic inputs comprised of a weighted bipartite graph and arrival probabilities $p_t \sim U(0, 1)$. We generate synthetic bipartite graphs drawn from the Erdős-Rényi (ER) (Erdős and Rényi, 1960), Barabási-Albert (BA) (Albert and Barabási, 2002), and geometric (b-RGG) random graph families. To simulate performance on real-world crowdsourcing tasks, we generate semi-synthetic graphs from OSMnx (Boeing, 2017), a library that encodes road networks for use in ride-sharing applications, and the gMission dataset (Chen et al., 2014), whose base graph comes from crowdsourcing data for assigning workers to tasks. Details on bipartite graph generation are in Appendix B.2.

In terms of notation, we say a bipartite graph has shape $(|L| \times |R|)$ if it has $|L|$ offline nodes and $|R|$ online nodes. A graph *configuration* refers to a graph family and fixed set of generating parameters (e.g., ER with $p = 0.5$).

Evaluation. Given an input graph G, a matching model M outputs a sequence of matching decisions as the arrival $\boldsymbol{a}_t \in \{0, 1\}$ of each online node t is revealed. We evaluate the performance of M on G by taking the average *competitive ratio* over ℓ realizations of the arrival vector $\boldsymbol{a}$:

$$\text{CR}(M, G) = \frac{1}{\ell} \sum_{j=1}^{\ell} \frac{M(G, \boldsymbol{a}^{(j)})}{\text{OPT}(G, \boldsymbol{a}^{(j)})}.$$

Here $M(G, \boldsymbol{a})$ is the weight of the matching returned by M on graph G with arrival sequence $\boldsymbol{a}$, and $\text{OPT}(G, \boldsymbol{a})$ is the weight of the max-weight matching in G based on *a priori* knowledge of $\boldsymbol{a}$. Similarly, we evaluate the performance of M over a graph configuration $\mathcal{G}$ by averaging the input-wise competitive ratio $\text{CR}(M, G)$ over many input graphs

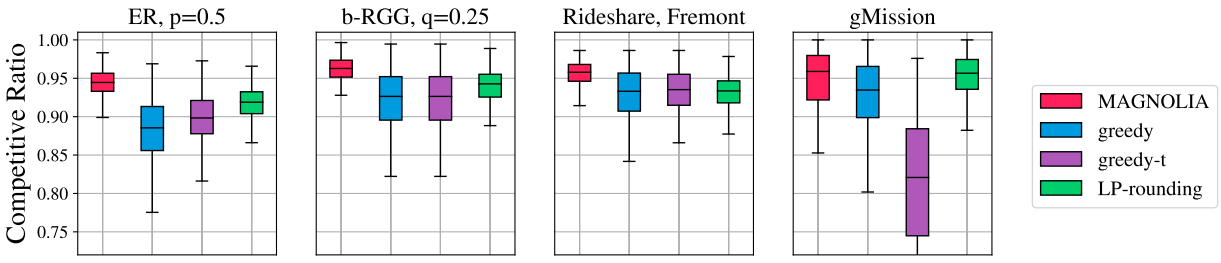

Figure 2. Boxplot showing the distribution of competitive ratios for MAGNOLIA and baselines across graph configurations. All graphs are of size (10×20), and results for additional configurations are available in Appendix B.8.1.

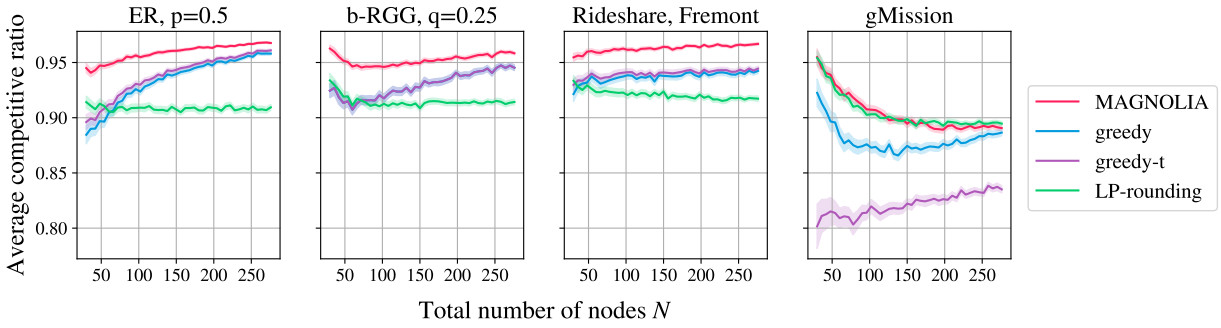

Figure 3. Evolution of competitive ratio over graphs of increasing size for a GNN trained on graphs of size (6×10). All test graphs have a 2:1 ratio of online to offline nodes. Because the threshold t for `greedy-t` is selected from a validation set over diverse graph configurations, it does not perform equally well on all configurations. Results for additional graph configurations are in Appendix B.8.2

$G \sim \mathcal{G}$. All competitive ratios are computed with respect to the offline optimal algorithm OPT, since computing OPT_{on} is intractable for graphs of reasonable size.

Baselines. We compare our learned models to several strong baselines. Upon the arrival of an online node, `greedy` picks the maximum weight available edge. To better trade off between short and long-term rewards, `greedy-t` (Alomrani et al., 2022), makes the same decision as `greedy` if the maximum edge value is above some threshold t, and skips otherwise. This threshold parameter is tuned to maximize the average competitive ratio over a diverse validation set. Finally, `LP-rounding` is an OBBM approximation algorithm by Braverman et al. (2022), which achieves an approximation ratio of 0.632 and outperforms current best-published approximation algorithm (Naor et al., 2023) in practice.

To augment the potential value of MAGNOLIA in practice, a central tenant in its design is that it should demonstrate strong out-of-distribution generalization to unseen graph configurations. For example, to obtain strong performance on ER graphs of a certain density, MAGNOLIA should not need to be trained on ER graphs with that density. As such, because the approach by Alomrani et al. (2022) learns a tailored matching policy per graph configuration, we do not consider it as a baseline. Moreover, it is difficult to make a fair comparison—as an RL-based approach, Alomrani et al. (2022) requires more training data, larger training graph sizes, and more compute time than MAGNOLIA to learn an effective policy. Thus, any comparison with a shared training set would be ill-suited to their method.

4.3. Results

MAGNOLIA makes good decisions. We train the GNN underlying MAGNOLIA on a collection of 2000 OBBM instances from 3 graph configurations, then evaluate performance on 6000 unseen instances from a broader selection of 12 configurations. Figure 2 shows the distribution of competitive ratios for a representative sample of configurations. On these and others, MAGNOLIA consistently achieves an average CR that is 2-5% higher than the best baseline.

MAGNOLIA shows size generalization. An important characteristic of GNNs trained for combinatorial tasks is the extent to which they generalize to graphs of larger size. Despite being trained exclusively on 16-node graphs, we see in Figure 3 that MAGNOLIA's performance largely remains consistent for inputs that are up to 18 times larger. For

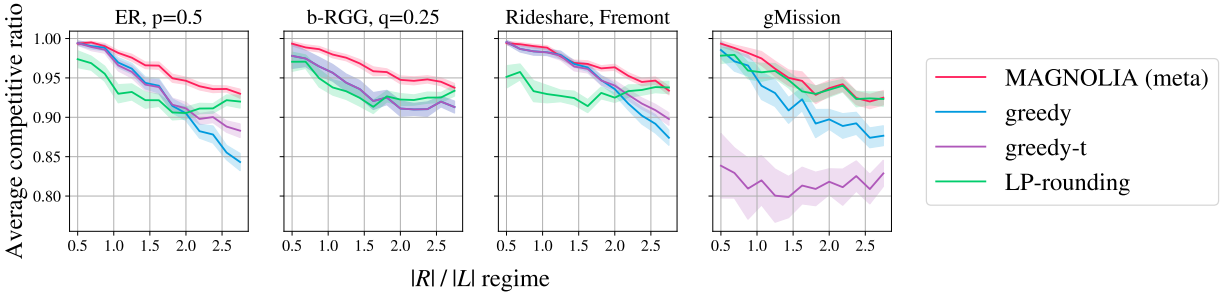

Figure 4. Evolution of competitive ratio over regimes for MAGNOLIA enabled with a meta-GNN. For evaluation, $|L|$ is kept fixed at 16 offline nodes, and $|R|$ varies from 8 to 45 online nodes. Results for additional graph configurations are available in Appendix B.8.4.

the single graph configuration where performance degrades as a function of graph size, the decline is in line with that of `LP-rounding`, whose approximation guarantees are invariant to graph size.

Meta-models can improve regime generalization. We observe that the ratio $|R|/|L|$ of online to offline nodes in an input materially impacts the performance of OBBM algorithms. In light of this, we also study how MAGNOLIA's performance generalizes across ratios $|R|/|L|$, or *regimes*. Though MAGNOLIA performs well on regimes similar to those its GNN was trained on, this does not translate to ones with different dynamics; a model trained in an "offline-heavy" regime ($|R|/|L| < 1$) tends to perform poorly in an "online-heavy" regime ($|R|/|L| > 1$), and conversely.

A natural solution is to replace MAGNOLIA's GNN with a meta-model that selects between multiple GNNs—each trained on different regimes—on an instance-by-instance basis. In our experiments, we use a meta-GNN that selects between two GNNs trained on graph sizes (10×6) and (6×10) based on the predicted competitive ratio. Figure 4 gives regime generalization results for MAGNOLIA for this meta-model. We find that the meta-GNN recovers a simple configuration-dependent threshold rule on $|L|/|R|$. See the ablation tests in Appendix B.8.4 for a comparison against a purely threshold-based meta-model. Our model consistently outperforms greedy baselines, and while `LP-rounding` is eventually better in very online-heavy regimes which are out-of-distribution for the GNN, MAGNOLIA performs especially well in more balanced regimes where `LP-rounding` is worst.

MAGNOLIA is robust to noisy inputs. One criticism of the OBBM model is that exact knowledge of both the underlying graph and arrival probabilities is impractical. Instead, it is often more reasonable to assume access to noisy estimates of these values coming from data or an ML model. A key question, then, is how robust these OBBM algorithms are to training and testing on noisy input. To test this, we run several experiments with varying levels of noise: for level ρ, $\mathcal{N}(0, \rho^2)$ noise is added independently to each edge weight w_{ij} and each arrival probability p_t. Figure 5 shows how the performance of MAGNOLIA and baselines degrades as a function of ρ— we see that while robustness to noise is configuration-dependent, MAGNOLIA is consistently the most robust of the models we consider.

The extent to which performance degrades for a particular graph configuration is related to the variance of edge weights incident on each online node. Adding noise increases the frequency with which an algorithm makes sub-optimal decisions, and so if this variance is high, mistakes are especially costly. This helps to explain the sizeable decline in competitive ratio for ER and Rideshare graphs that we observe in this experiment: the variance of edge weights incident on an online node is an order-of-magnitude larger in Erdős-Rényi and Rideshare graphs than in b-RGG and gMission graphs.

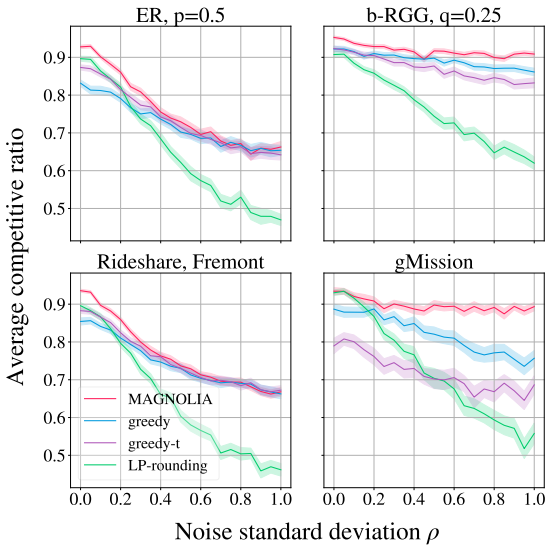

Figure 5. Evolution of competitive ratio as a function of noise level ρ for graphs of size (10×30). Results for additional graph configurations are available in Appendix B.8.5.

5. Conclusions

In this paper, we studied online matching in digital marketplaces, a problem of critical importance across various sectors such as advertising, crowdsourcing, ridesharing, and kidney exchange. We focused on the Online Bayesian Bipartite Matching problem, introducing a novel approach using GNNs to approximate the online optimal algorithm, which maximizes the expected weight of the final matching under uncertain future demand. We provided theoretical guarantees demonstrating that the value-to-go function can be efficiently approximated in bipartite random geometric graphs through local information aggregation—a process well-suited to GNNs. Empirically, our GNN-based algorithm, MAGNOLIA, achieved strong performance against state-of-the-art baselines, showcasing strong generalization across different problem sizes and graph families.

This work exposes many directions for future research. For example, can we use the theoretical framework from Section 3 to analyze the performance of GNNs on other problems beyond matching? In this vein, b-RGGs are one example of a graph family that exhibits locality and decomposibility. What other graph families exhibit these properties?

Impact statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgements

This work was supported in part by NSF grant CCF-2338226, AFSOR grant FA9550-23-1-0251, ONR grant N000142212771, and a National Defense Science & Engineering Graduate (NDSEG) fellowship.

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. 4.2, 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, 4.2, 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. 4.2, 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, 4.2

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *CoRR*, abs/2105.14491, 2021. URL https://arxiv.org/abs/2105.14491. B.7

Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24 (130):1–61, 2023. 1.2

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. 4.2

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. 4.2, 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. 3.1

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

Pengfei Li, Jianyi Yang, and Shaolei Ren. Learning for edge-weighted online bipartite matching with robustness guarantees. In *International Conference on Machine Learning (ICML)*, 2023. 1.2

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. 3.1, 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. *CoRR*, abs/1810.02244, 2018. URL http://arxiv.org/abs/1810.02244. B.7

Joseph Naor, Aravind Srinivasan, and David Wajc. Online dependent rounding schemes. *ArXiv*, abs/2301.08680, 2023. 1.2, 4.2

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. 3.8

Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29:217–250, 2020. 1, 1.1, 3

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 $\boldsymbol{x_1}, \ldots, \boldsymbol{x}_N \in [0,1]^d$, $\varepsilon > 0$, $\Delta \leq \frac{\varepsilon}{2d}$, 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 3.5. *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. □

Corollary 3.6. *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 3.5. We note that $\frac{\log N}{\log \log N} = O(\log N)$. □

Theorem 3.7. *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 3.6 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. □

Lemma 3.10. *For any $G \in \text{supp}(G(m, n, \mathcal{D}, \Delta)$ and any hypercube partition π, $\mathcal{V}(G(\pi)) \leq \mathcal{V}(G)$.*

Proof. 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. □

Lemma 3.11. *For any $G \in \text{supp}(G(m, n, \mathcal{D}, \Delta))$, $\varepsilon > 0$, and $k = \lceil \frac{\varepsilon}{2d\Delta} \rceil$, $\mathbb{E}_{\pi \sim \Pi_k}\left[\mathcal{V}(G(\pi))\right] \geq (1 - \varepsilon)\mathcal{V}(G)$.*

Proof. 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 $\boldsymbol{a} \in \{0, 1\}^m$,

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

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

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

□

Lemma 3.12. *Let $\mathcal{D}$ be a β-smooth distribution and $\Delta = O(N^{-1/d})$ where N is sufficiently large. For all $\varepsilon \in (0, 1/2]$, 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 the draw of $\ell \geq \frac{2}{\varepsilon^2} \log \frac{4}{\delta}$ 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 3.7 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 3.10. 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

$$\begin{aligned}
\Pr \left[\left| S_\ell - \mathbb{E}[S_\ell] \right| \geq \varepsilon \mathbb{E}[S_\ell] \right] &\leq \Pr \left[\left| S_\ell - \mathbb{E}[S_\ell] \right| \geq \varepsilon(1-\varepsilon) \mathcal{V}(G) \right] && \text{Lemma 3.11} \\
&\leq 2 \exp \left(- \frac{2\varepsilon^2 (1-\varepsilon)^2 \mathcal{V}(G)^2}{\ell \cdot \mathcal{V}(G)^2 / \ell^2} \right) \\
&\leq 2 \exp \left(-\ell \varepsilon^2 / 2 \right) \\
&\leq \delta / 2.
\end{aligned}$$

Thus for sufficiently large N, we've shown that

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

□

Theorem 3.13. *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 $\delta\in(0,1]$.*

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 3.11 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 $G\in\text{supp}(G(m,n,\mathcal{D},\Delta)$. 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 3.12 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 6, 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., 2018; Brody et al., 2021). 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	$[1e-5, 1e-10]$

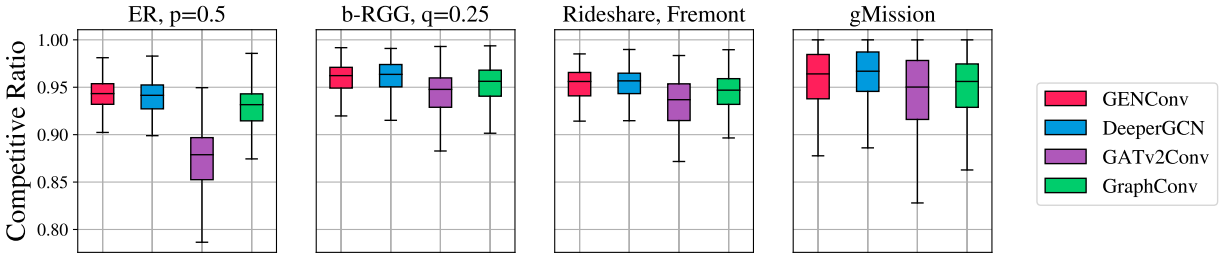

Figure 6. Boxplot showing the distribution of competitive ratios for MAGNOLIA with different underlying GNN architectures across graph configurations. All graphs are of size (10×20).

B.8. Complete results for Section 4.3

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

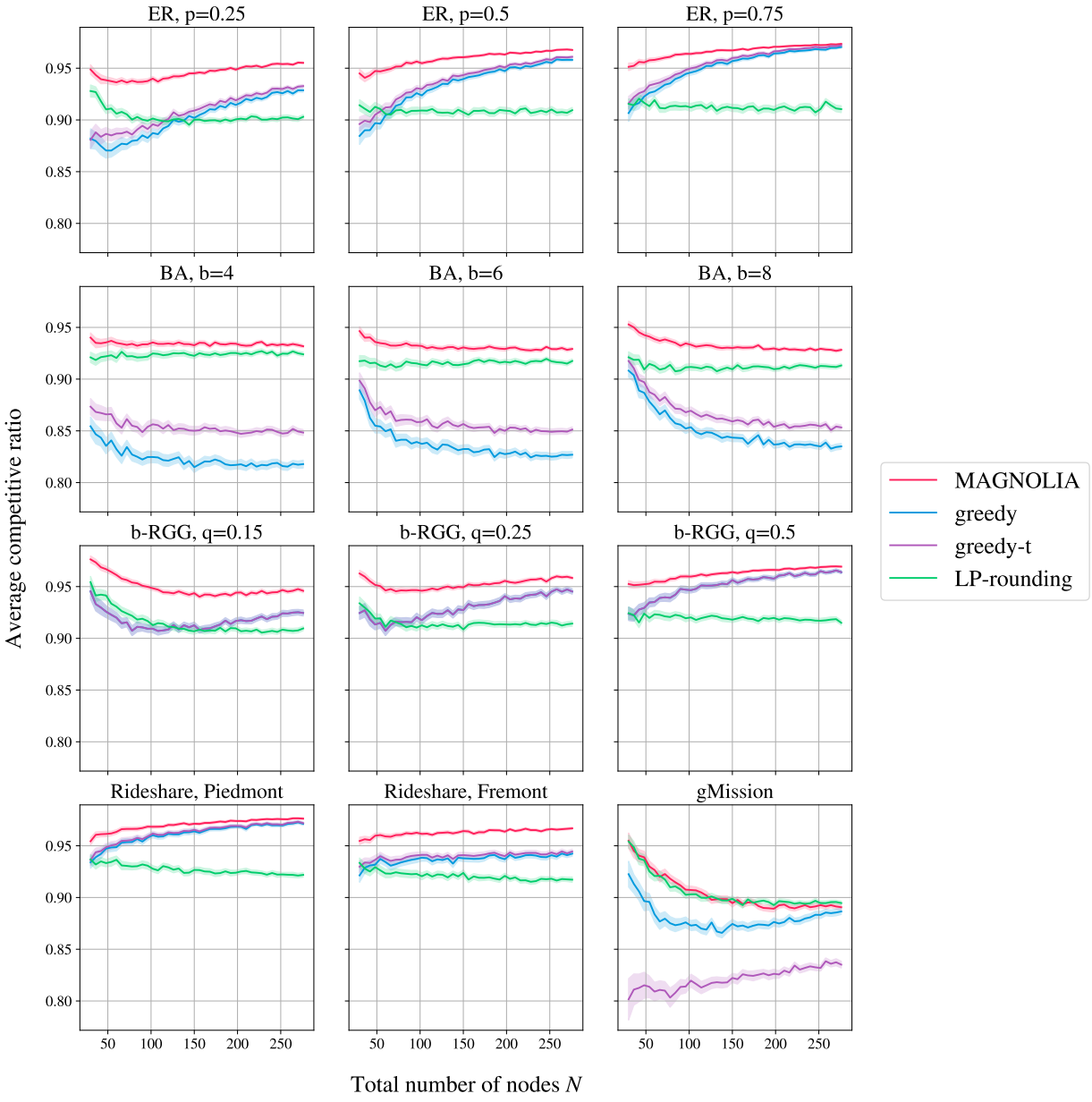

Figure 7. Evolution of competitive ratio over graphs of increasing size for a GNN trained on graphs of size (6×10). All test graphs have a 2:1 ratio of online to offline nodes.

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 8 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.

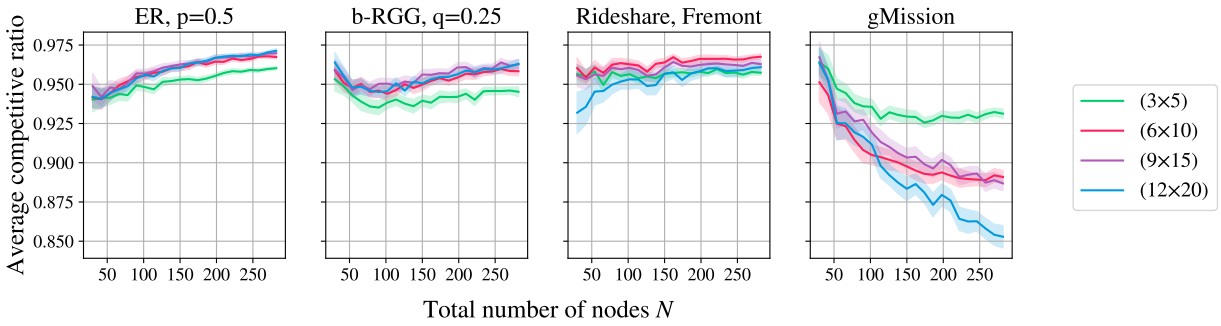

Figure 8. Evolution of competitive ratio over graphs of increasing size for GNNs trained on graphs of different sizes with the same (6:10) ratio. All test graphs have a 1:2 ratio of offline to online nodes.

B.8.4. META-MODEL IMPROVES REGIME GENERALIZATION

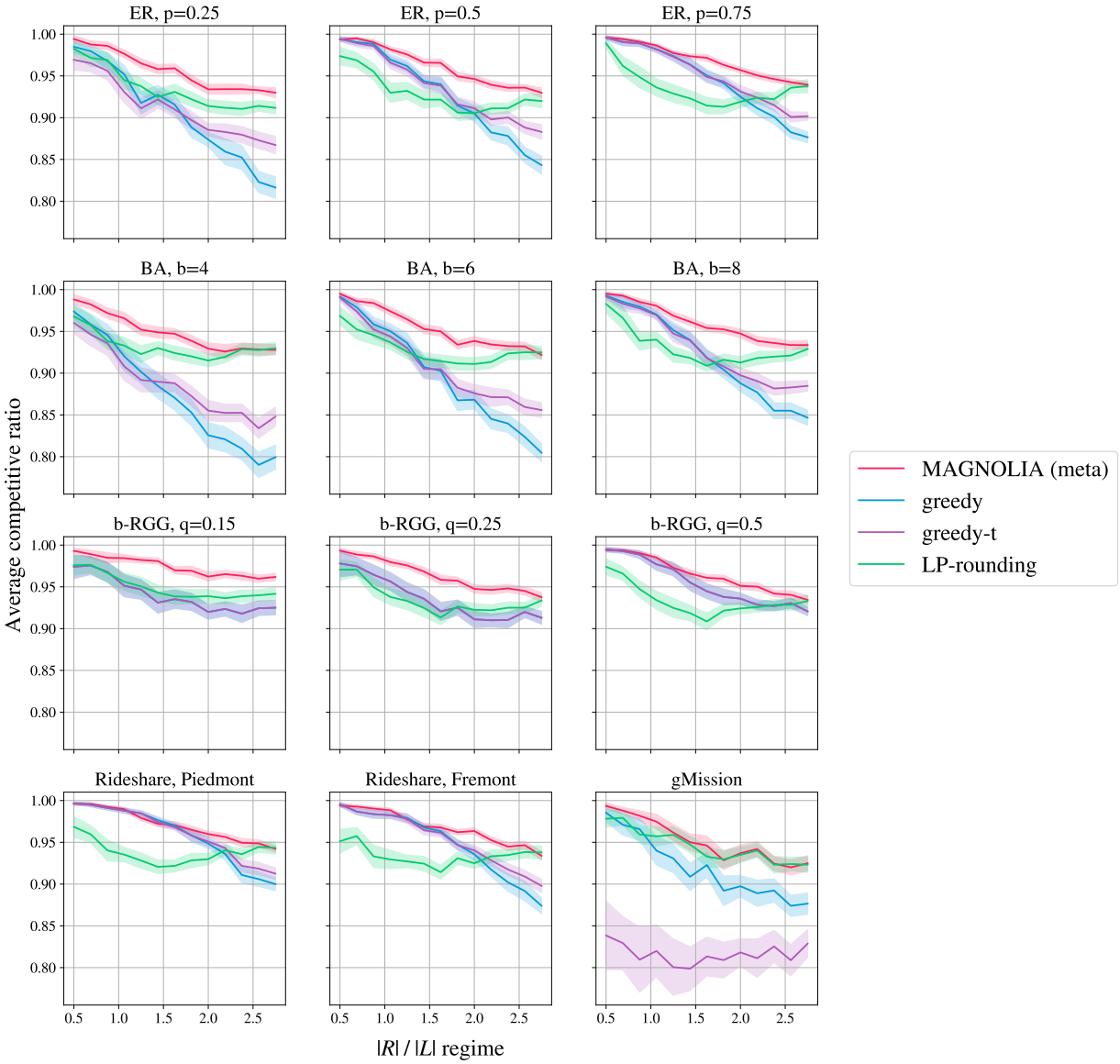

Figure 9. Evolution of competitive ratio over regimes for MAGNOLIA enabled with a meta-GNN. For evaluation, $|L|$ is kept fixed at 16 offline nodes, and $|R|$ varies from 8 to 64 online nodes

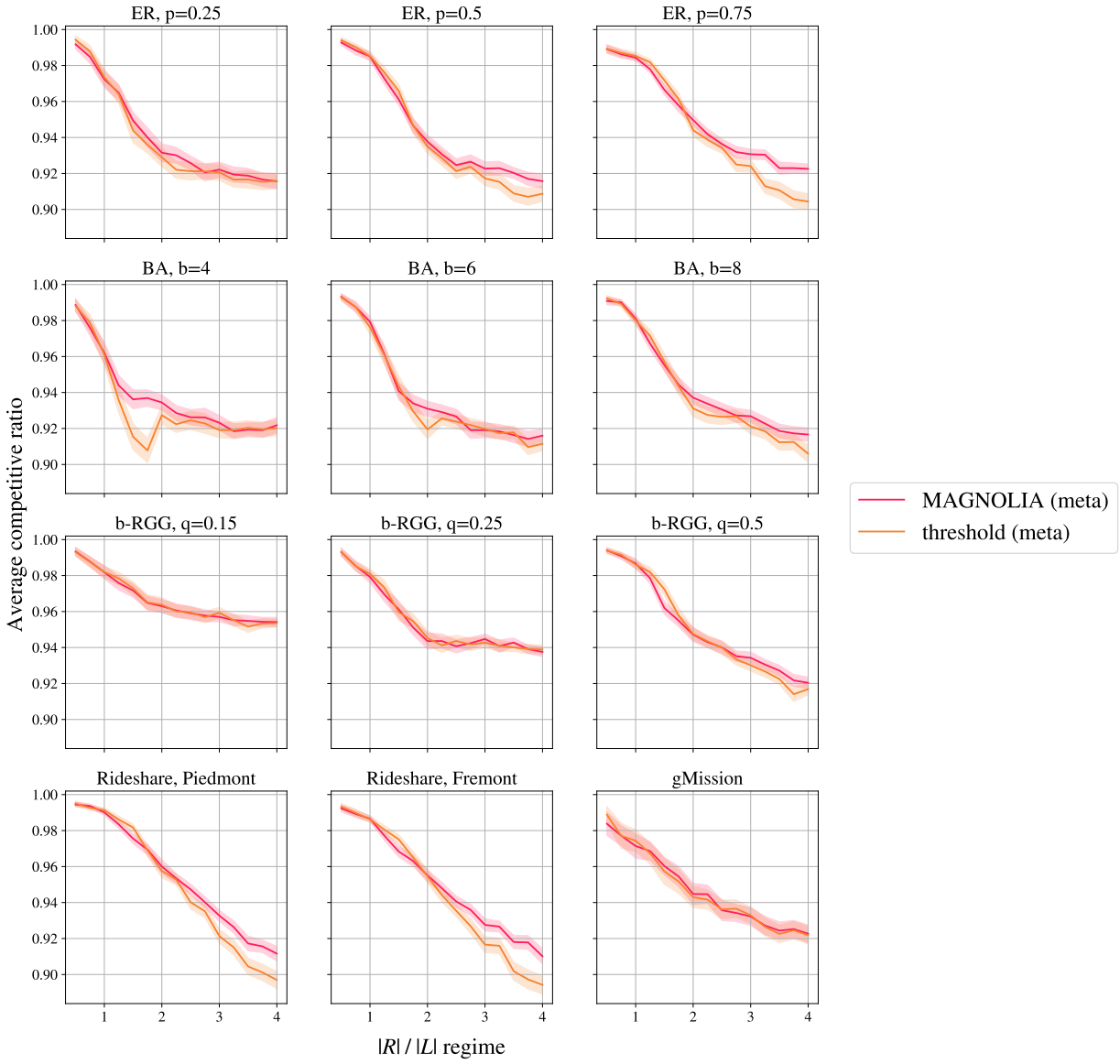

Figure 10. Evolution of competitive ratio over regimes for MAGNOLIA enabled with a meta-GNN against simple threshold model. For evaluation, $|L|$ is kept fixed at 16 offline nodes, and $|R|$ varies from 8 to 64 online nodes

B.8.5. MAGNOLIA IS ROBUST TO NOISY INPUTS

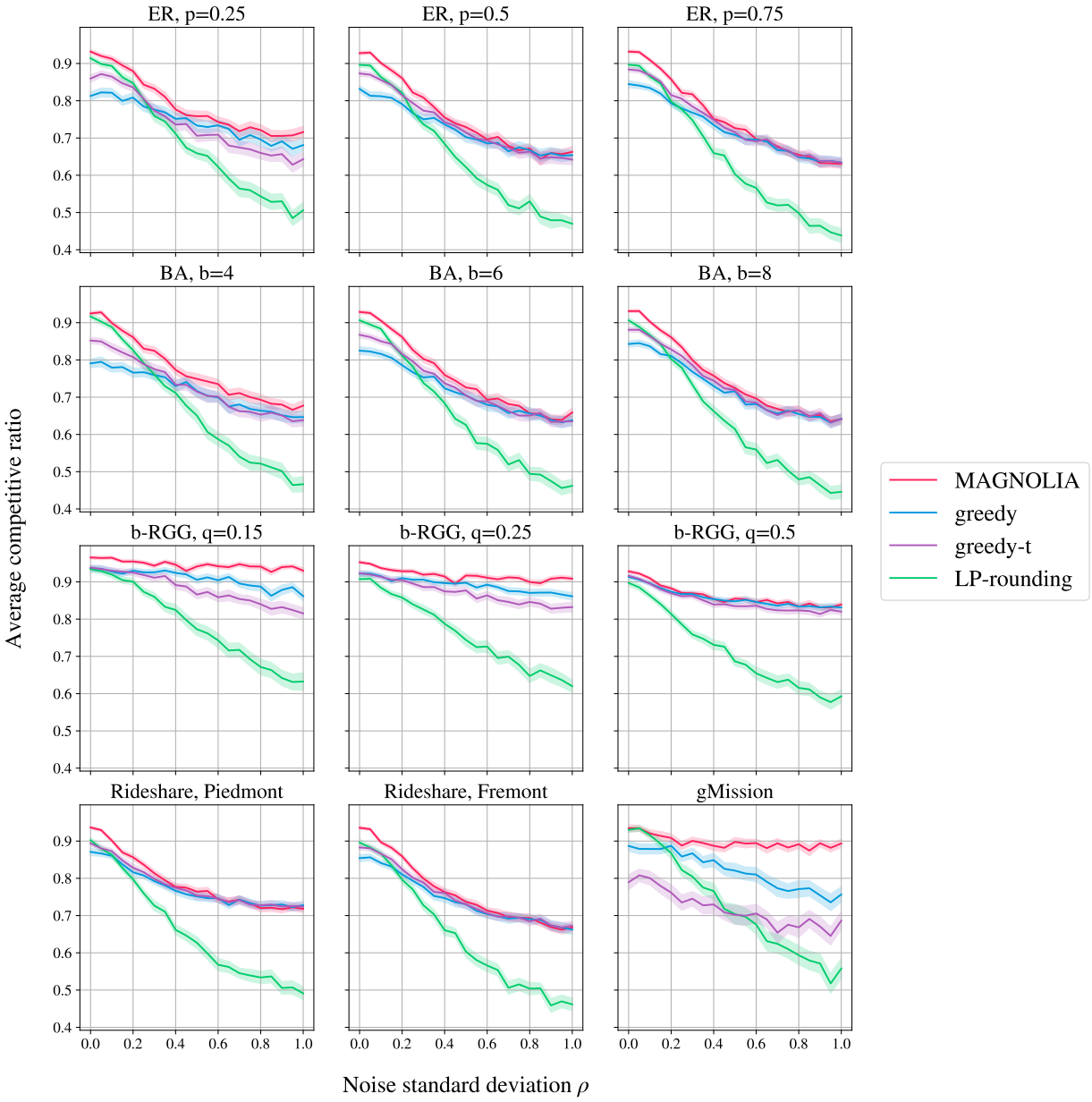

Figure 11. Evolution of competitive ratio as a function of noise level ρ for graphs of size (10×30). A $\mathcal{N}(0, \rho^2)$ noise is added independently to each edge weight and arrival probability.