arXiv:2601.04423v2 [cs.DS] 19 Nov 2026

Learning Multinomial Logits in $O(n \log n)$ time

Flavio Chierichetti*
Reddit
flavio.chierichetti@gmail.com

Mirko Giacchini*
Columbia University, Italian Academy Fellow
mg5011@columbia.edu

Ravi Kumar
Google Research, Mountain View
ravi.k53@gmail.com

Silvio Lattanzi
Google Research, Barcelona
silviol@google.com

Alessandro Panconesi
Sapienza University of Rome
ale@di.uniroma1.it

Erasmo Tani*
Cornell University
et484@cornell.edu

Andrew Tomkins
Google Research, Mountain View
atomkins@gmail.com

Abstract

A Multinomial Logit (MNL) model is composed of a finite universe of items $[n] = \{1, \ldots, n\}$, each assigned a positive weight. A query specifies an admissible subset—called a *slate*—and the model chooses one item from that slate with probability proportional to its weight. This query model is also known as the *Plackett–Luce model* or *conditional sampling* oracle in the literature. Although MNLs have been studied extensively, a basic computational question remains open: given query access to slates, how efficiently can we learn weights so that, for *every* slate, the induced choice distribution is within total variation distance ε of the ground truth? This question is central to MNL learning and has direct implications for modern recommender system interfaces.

We provide both adaptive and non-adaptive algorithms for this task. Our first adaptive algorithm makes $O(\frac{n \log n}{\varepsilon^2})$ queries and learns every choice distribution in total variation distance ε. We also provide a second adaptive algorithm that with $O(\frac{n \log n}{\varepsilon^3})$ queries provides a stronger guarantee: each choice distribution is approximated to a multiplicative $1 \pm \varepsilon$ error (except for subsets of mass smaller than ε). We complement these algorithms with lower bounds of $\Omega(\frac{n \log n}{\varepsilon^2})$ for total variation distance and $\Omega(\frac{n \log n}{\varepsilon^3})$ for the multiplicative error guarantee, showing the optimality of our adaptive algorithms.

Our non-adaptive algorithm makes $O\left(\frac{n^2}{\varepsilon^3} \log n \log \frac{n}{\varepsilon}\right)$ queries to obtain guarantees under the multiplicative error notion. We complement this result with an $\Omega(\frac{n^2 \log n}{\varepsilon^2})$ lower bound for any non-adaptive algorithm, showing that our algorithm is tight in its dependence on the support size n, within a $\log n$ factor.

All algorithms query only slates of size two and run in time proportional to their query complexity.

*Work done in part while at Sapienza University of Rome.

Contents

1 Introduction **1**

2 Related Work **4**

3 Technical Preliminaries **6**

4 Overview of Results and Techniques **7**
- 4.1 Learning MNLs Adaptively in d_{mul}-Error . . . 7
- 4.2 Learning MNLs Adaptively in d_1-Error . . . 10
- 4.3 Learning MNLs Non-Adaptively . . . 11
- 4.4 Lower Bounds . . . 12

5 Algorithmic Primitives **13**

6 Adaptive Algorithm for d_{mul} Error **14**
- 6.1 Constructing a Cluster Graph with $O\left(\frac{n \log n}{\varepsilon^2}\right)$ Queries . . . 14
- 6.2 Constructing an Estimation-Forest with $O\left(\frac{n \log n}{\varepsilon^3}\right)$ Queries . . . 17
- 6.3 Learning the MNL from the Estimation-Forest . . . 23

7 Adaptive $O(\frac{n \log n}{\varepsilon^2})$ Algorithm for d_1 **27**
- 7.1 Constructing an ε-Weak-Estimation-Forest . . . 27
- 7.2 Learning the MNL from a Weak-Estimation-Forest . . . 31

8 The Non-Adaptive Algorithm **32**
- 8.1 Spreading the Queries Among the Cluster Items . . . 33
- 8.2 Computing an MNL: Obtaining an Estimation-Forest . . . 39

9 Lower Bounds **45**
- 9.1 Lower Bound for Approximate Coin Selection . . . 46
- 9.2 Lower Bounds for Adaptive Algorithms . . . 47
- 9.3 Lower Bounds for Non-Adaptive Algorithms . . . 49

10 Conclusions and Open Problems **51**

A Missing Proofs for Section 5 **52**

B Missing Proofs for Section 6: Computing Approximate Orderings **55**

C Missing Proofs for Section 7 **57**

D Missing Proofs for Section 8 **59**

E Missing Proofs for Section 9: Extension to Pseudo-MNLs **59**

F An Additive Approximation for Pairs is not Sufficient **65**

1 Introduction

Multinomial Logit models (MNLs), also known as softmax or Plackett–Luce models, are widely used to model choice behavior in machine learning and economics. They describe winning distributions over alternatives parameterized by item weights: given a universe U of items, an MNL M assigns each $i \in U$ a weight w_i, and for any non-empty subset $S \subseteq U$, defines $M_S(i) = w_i / \sum_{j \in S} w_j$ as the probability of selecting i from S. Such models underlie diverse applications, from token prediction in large language models to content selection in recommender systems, where they capture how preferences depend on the available slate.

Most prior work focuses on estimating MNL parameters or the induced distribution on the universal slate $S = U$, which suffices for identifying the globally most preferred items or obtaining a top-k ranking. In contrast, we address the more challenging task of approximating the MNL distribution for *all* slates, motivated by practical needs in modern recommender systems. Consider a platform such as Netflix offering choices of movies. It is now broadly understood that simply displaying a very long list of top titles does not provide a compelling user experience. Instead, these platforms define a large and rapidly changing number of relevant subsets of the entire movie catalog: action movies, foreign movies, movies similar to a particular anchor movie the user recently watched, and so forth. The interface then shows a sequence of carousels, perhaps a carousel of "top movies" followed by "movies similar to Ponyo" then "new arrivals", each one ordered to show the user's best options from the class. To drive such an interface, it is important to approximate the winning distribution for *every* one of these subsets simultaneously, to be ready to display it when needed. As the possible subsets of interest are constantly updated by the platform, it is critical to approximate the MNL's output on all possible subsets $S \subseteq U$.

Furthermore, for a particular subset, such as that containing all action movies, the platform will not show a single option, but will instead show a carousel with a moderate number of suggestions. While some previous work focused exclusively on ranking the items, practical recommender systems require scoring them for at least two key reasons. First, the number of items shown should depend on their scores: if there are four high-scoring movies, it might be better to only display those, rather than adding the next six, which may have very little chance of being selected. Second, the interface might have more richness than just the carousel itself. For instance, if the top movie of a carousel has a much higher score than the next one, the platform might feature this movie more prominently, for example, by using a specialized rendering or by allocating more space to it. Hence, it is crucial to obtain estimates of the weights that provide accurate winning distributions on all slates.

MNL and MNL Learning. A *multinomial logit (MNL) model* supported on the universe $U = [n] = \{1, \ldots, n\}$ is specified by a set $\{w_1, \ldots, w_n\}$ of n positive values called *weights*. A *slate* is a non-empty subset of $[n]$. An MNL M, for any given slate $S \subseteq [n]$, induces a conditional distribution denoted M_S whose support is S and where the probability of each item $i \in S$ is given by:[1]

$$M_S(i) = \frac{w_i}{\sum_{j \in S} w_j}.$$

An MNL M can be accessed by a Sample oracle, which operates as follows: given a slate S, $\mathsf{Sample}(S)$ returns $i \in S$ chosen according to the distribution M_S. Given MNLs M and M', we define two notions of distance between them:

$$d_\infty(M, M') := \max_{\substack{S \subseteq [n] \\ S \neq \varnothing}} \|M_S - M'_S\|_\infty \quad \text{and} \quad d_1(M, M') := \max_{\substack{S \subseteq [n] \\ S \neq \varnothing}} \|M_S - M'_S\|_1.$$

[1] The terminology we adopt comes from the Economics literature (Train, 2003), this is called a *logit* model because if we let $w_i = e^{\theta_i}$, then $M_S(i) = \mathrm{softmax}(S)_i = e^{\theta_i} / \sum_{j \in S} e^{\theta_j}$.

We will also consider a multiplicative notion of error. Specifically, we say that $d_{\text{mul}}(M, M') \leq \varepsilon$ if for each slate $S \subseteq [n]$ there exists $S' \subseteq S$ such that:

$$(1-\varepsilon) \cdot M_S(i) \leq M'_S(i) \leq (1+\varepsilon) \cdot M_S(i) \quad \text{for each } i \in S',$$
$$\sum_{i \in S \setminus S'} M_S(i) \leq \varepsilon,$$
$$\sum_{i \in S \setminus S'} M'_S(i) \leq \varepsilon$$

That is, M' provides multiplicative $1 \pm \varepsilon$ approximation of the winning distributions, except for sub-slates that have total mass of at most ε. Note that $d_1(M, M') \leq 3 \cdot d_{\text{mul}}(M, M')$, and therefore learning an MNL in d_{mul} error directly implies learning for d_1 (and d_∞). This type of multiplicative guarantee is also common in the distribution testing literature (Canonne et al., 2015).

In this paper we obtain algorithms that approximate an unknown MNL M in d_{mul} error and d_1 distance, while our lower bounds apply even to the less challenging problem of obtaining estimates with small d_∞ distance.

Definition 1 (MNL Learning Problem)**.** Given Sample oracle access to an MNL M and an $\varepsilon \in (0,1)$, the *MNL learning problem for* d_1 (resp. d_{mul}, and d_∞) is to output an MNL $\hat{M}$ such that $d_1(M, \hat{M}) \leq \varepsilon$ (resp. $d_{\text{mul}}(M, \hat{M}) \leq \varepsilon$, and $d_\infty(M, \hat{M}) \leq \varepsilon$). The MNL produced in output is represented using the logarithms of its weights.[2]

Main Results. In this paper, we study algorithms for the MNL learning problem. We obtain three algorithms, two using adaptive queries and the other using non-adaptive queries. All algorithms query only slates of size two and run in time proportional to their query complexity. Our adaptive algorithm for d_{mul} makes $O\left(\frac{n}{\varepsilon^3} \log n\right)$ queries while our adaptive algorithm for d_1 makes $O\left(\frac{n}{\varepsilon^2} \log n\right)$ queries. We give lower bounds matching our algorithmic results up to constant factors: we give lower bounds of $\Omega(\frac{n \log n}{\varepsilon^3})$ for d_{mul}-error and $\Omega(\frac{n \log n}{\varepsilon^2})$ for d_∞-error (and therefore, also d_1). Summarizing:

Theorem 2 (Informal)**.** *For any constant $\varepsilon > 0$, the complexity of learning an MNL within d_{mul}-error ε by making* Sample *queries adaptively is $\Theta(n \log n)$.*

Our non-adaptive algorithm makes $O\left(\frac{n^2}{\varepsilon^3} \log n \log \frac{n}{\varepsilon}\right)$ queries to learn an MNL in d_{mul}-error; this is complemented by a lower bound of $\Omega(\frac{n^2}{\varepsilon^2} \log n)$ for d_∞-error. Summarizing:

Theorem 3 (Informal)**.** *For any constant $\varepsilon > 0$, the complexity of learning an MNL within d_{mul}-error ε by making* Sample *queries non-adaptively is between $O(n^2 \log^2 n)$ and $\Omega(n^2 \log n)$.*

Our results are surprising: our seemingly harder problem can be solved as fast as (noisy) sorting. Furthermore, our lower bounds hold for unit-time oracle queries of any slate size. Hence, restricting the algorithms to slates of size two incurs no loss in efficiency.

[2]Representing an MNL using the logarithms of its weights is standard in the ML and economics community (Seshadri et al., 2020; Train, 2003). Moreover, with a full representation of the $\hat{w}_i$'s, the weights could require $\Omega(n^2)$ bits just to be stored. For instance, consider the MNL on $[n]$ with weights $w_i = 2^i$. Since $\frac{w_{i+1}}{w_i + w_{i+1}} = \frac{2}{3}$, any MNL $\hat{M}$ solving the MNL learning problem must satisfy $\hat{w}_{i+1} \geq \hat{w}_i \cdot (2 - 9\varepsilon)$. Therefore, each weight $\{\hat{w}_{n/2}, \ldots, \hat{w}_n\}$ requires $\Omega(n)$ bits for a total of $\Omega(n^2)$ bits. Hence, requiring that an algorithm outputs the weights, rather than their logarithms, would rule out the possibility of constructing any algorithm that runs in time $o(n^2)$. Other compact representations of the weights are also possible.

Technical Challenges. Existing methods, especially ones that approximate the winning distribution over the universal slate $[n]$, do not seem to apply to our problem. As a simple example of the difficulty, consider an algorithm that guarantees an ℓ_1-estimate of the full slate distribution within an error of $\varepsilon \in (0, 1/2)$. Consider now the MNL on $\{1, 2, 3\}$ with weights $w_1 = 1-\varepsilon$, $w_2 = w_3 = \varepsilon/2$. Suppose the algorithm returns the estimate $\hat{w}_1 = 1-\varepsilon$, $\hat{w}_2 = \frac{3\varepsilon}{4}$, $\hat{w}_3 = \frac{\varepsilon}{4}$; clearly, $\|w-\hat{w}\|_1 = \frac{\varepsilon}{2} \leq \varepsilon$. But, $\left|\frac{w_2}{w_2+w_3} - \frac{\hat{w}_2}{\hat{w}_2+\hat{w}_3}\right| \geq \frac{1}{4}$, and therefore the algorithm cannot guarantee small error on the slate $\{2, 3\}$. Similarly, as we discuss in Appendix F, prior work that additively estimates the winning distributions on all size-two slates cannot be used to obtain a good approximation on every slate. Moreover, noisy sorting algorithms also do not imply our guarantees. Indeed, while our final algorithm will perform a noisy sorting step at the beginning, the key challenge lays in performing the additional comparisons that allow the accurate estimate of each conditional distribution, as we detail below.

It is not difficult to obtain a cubic time algorithm for our problem. Indeed, consider the naive algorithm that works as follows. For each pair $\{i, j\}$ of items in the universe, estimate w_i/w_j to within a $(1 \pm \varepsilon)$-multiplicative error, or declare that their ratio (or its inverse) is larger than $\frac{n}{\varepsilon}$. One can easily show that this algorithm will need to query each pair $\approx \frac{n \log n}{\varepsilon^3}$ times to guarantee these bounds; the total cost would then be $\approx \frac{n^3 \log n}{\varepsilon^3}$. From the output of this algorithm, it is easy to approximate the output distribution for any slate.[3]

With some effort, this algorithm can be improved. The idea is to carefully control the pairs of items, querying only pairs that are nearby in the order induced by the weights (which can be approximately inferred via a noisy sorting algorithm). To avoid querying too many nearby pairs, one can first cluster items whose weights are within a constant factor, in a query-efficient manner, and then select a center from each cluster. One can determine the weight ratio of each item to its cluster center, and the weight ratios of successive items in the sorted list of cluster centers. This method can be shown to produce an algorithm that compares $O(n)$ pairs of items, with each such comparison performing $\approx \frac{n}{\varepsilon^3} \log^3 n$ queries. While this yields a quadratic time algorithm, it is unclear how this can be further improved to being quasi-linear.

Overview of Methods. We focus on our algorithm for d_{mul}, which we find more intuitive to highlight our ideas. We construct our quasi-linear adaptive algorithm by building on the clustering idea described above. We first partition the universe into clusters of similar-weight items and select a center for each cluster. We then estimate the ratio of the weights w_i/w_c for every item i in the cluster with center c. Finally, we construct a forest on the cluster centers, where every edge is labeled with an estimate of the ratio between the weights of the two centers it connects. We call this data structure the *estimation-forest*. This allows us to obtain estimates of the ratio of the weights for arbitrary pairs of items by combining these estimates along paths in the forest.

Following this strategy, the error compounds multiplicatively along the paths. To circumvent this issue without requiring more accurate ratio estimates—which would lead to a higher complexity—we design the forest so that any two centers whose weights the algorithm might want to compare are at a short distance from each other. To achieve this property, the topology of the forest is constructed adaptively. Additionally, to further improve the sample complexity (and runtime), we

[3]Indeed, if the items of this slate have weights that are within a $\frac{n}{\varepsilon}$ factor of each other, the $(1 \pm \varepsilon)$-approximation error will make it possible to approximate the winning probability of any item to within a $(1 \pm O(\varepsilon))$-factor (so that the total variation error will be at most $O(\varepsilon)$). If, instead, the slate contains pairs $\{i, j\}$ of items such that $w_i/w_j < \frac{\varepsilon}{n}$, then the lighter item i will have a probability of winning in the slate not larger than $O(\frac{\varepsilon}{n})$, and hence we can estimate its winning probability to be zero—given that there are at most $n-1$ such light items in a slate, the total variation error is no larger than $O(\varepsilon)$.

dynamically adjust the number of queries required to approximate the weight ratio between two centers. In particular, if the total weight of items lighter than a given item i is not large enough with respect to the weight of i, then it becomes unimportant to estimate the ratio of the weight of i over the weight of any of these items. Our estimation-forest data structure dynamically determines query sequences for weight-ratio estimation and enables all cluster-center–cluster-center and cluster-item–cluster-center comparisons in $O\left(\frac{n}{\varepsilon^3}\log n\right)$ queries. To obtain a tight bound in ε for d_1-error, we relax the notion of estimation-forest and allow some edges to be approximated to less precision if this does not hurt the overall estimation.

While the above algorithm is adaptive, we also obtain a non-adaptive version. The idea is to first design a new adaptive algorithm that queries each pair of items only $O\left(\frac{1}{\varepsilon^3}\log n\log\frac{n}{\varepsilon}\right)$ times. We then query every pair a fixed number of times and then simulate this new adaptive algorithm on the precomputed answers; this leads to a non-adaptive algorithm with $O\left(\frac{n^2}{\varepsilon^3}\log n\log\frac{n}{\varepsilon}\right)$ queries.

The three lower bounds in our paper are proved using a reduction from the problem of identifying the k coins with the highest heads probability in a collection of n biased coins. In particular, we construct an MNL supported on an even-sized universe whose items are divided into pairs, each pair representing the two sides of a coin. We then order the pairs so that the weights of the items in a pair are much larger than those in preceding pairs. This way, we can assume without loss of generality that any learning algorithm is only querying slates corresponding to our original pairs.

Organization. In Section 2 we review related work. Section 3 introduces key tools and notation that we use throughout the paper. Section 4 gives a more detailed technical overview of our algorithms and techniques. In Section 5 we introduce two estimation primitives used by our main algorithms, which we present in the subsequent sections: in Section 6 we analyze our adaptive algorithm for d_{mul}-error, and then we relax the notion of estimation-forest to obtain a tight dependence in ε for d_1-error in Section 7, finally in Section 8 we present our non-adaptive algorithm. Section 9 contains the proofs of our adaptive and non-adaptive lower bounds. We conclude in Section 10 with several open questions.

All the proofs missing from the main body of the paper can be found in the appendix.

2 Related Work

The problem of learning MNLs on *all* slates from Sample queries arises naturally from several perspectives. Our work is related to, yet distinct from, the existing literature. First, prior work on MNL fitting has provided approximation guarantees only for the full slate or for pairs of items—both of which are strictly weaker than the guarantees we obtain. Second, our framework extends beyond classical MNL ranking and selection by capturing quantitative relationships among items, revealing how much and where certain items dominate, while preserving the $O(n\log n)$ efficiency of the best known ranking algorithms. Third, our problem can be viewed as a natural strengthening of distribution learning under conditional sampling, extending the "testing by learning" paradigm to recover all conditional distributions simultaneously in a more expressive and challenging setting. Finally, our results also strengthen prior work on learning Random Utility Models (RUMs), specialized to the MNL case. We elaborate on these connections below.

MNL Fitting. A large body of the literature focuses on finding MNL weights maximizing the likelihood of a collected dataset. In this setting, usually, the queries are either fixed (Zermelo, 1929; Ford Jr, 1957; Dykstra, 1960; Newman, 2023) or sampled from a distribution (Olesker-Taylor and Zanetti, 2024; Negahban et al., 2012, 2017; Maystre and Grossglauser, 2015). When the dataset

actually comes from a hidden MNL model, some of these algorithms guarantee that the estimated (normalized) weights approximate the hidden (normalized) weights (Negahban et al., 2012, 2017; Maystre and Grossglauser, 2015; Seshadri et al., 2020; Shah et al., 2016; Seshadri et al., 2019).

These works are not directly applicable to our setting because of the following two main issues. (i) Approximately recovering the normalized weights is equivalent to providing a good estimate of the winning distribution of the full slate. However, this is insufficient to accurately estimate the winning distribution for smaller slates, as we discussed in the Introduction. (ii) Most of these works assume that the maximum ratio between two weights is upper bounded by a constant (Negahban et al., 2012). Note that such an assumption would greatly simplify our problem given that we could accurately estimate the weights with respect to any anchor item. Therefore, the interesting setting is one where there is no *a priori* bound on the ratio of the weights. Furthermore, as these algorithms are non-adaptive, they are subject to our lower bound of $\Omega(n^2 \log n)$ queries for our problem.

Stepping outside the task of fitting the weights themselves, Falahatgar et al. (2018) adaptively query $O\left(\frac{n \cdot \log(n) \cdot \min\{n, 1/\varepsilon\}}{\varepsilon^2}\right)$ slates of size two (i.e., pairs) and produce an *additive* estimate within ε for all other *pairs*, in a family of models that are more powerful than MNLs. However, in order for an additive approximation of the slates of size two to generalize to all other slates with an ℓ_∞-error of ε', one needs $\varepsilon \leq \frac{\varepsilon'}{n}$ (proved in Section F). Therefore, applying their algorithm as a blackbox would require $\Omega(n^4 \log n)$ queries. Moreover, their algorithm heavily relies on providing an additive approximation—specifically, each estimated probability is rounded to the closest multiple of ε. Hence, it appears hard to generalize their work to larger slates even in a non-blackbox manner.

Other works considered minimizing the ℓ_∞-difference of the logarithm of the weights, or approximating the weight ratios of all the pairs (Yan et al., 2012; Han et al., 2020; Simons and Yao, 1999; Li et al., 2022). These guarantees would be sufficient to obtain a good estimate for all slates, but are not necessary. Indeed, if $w_1 = e^{1000}$, $w_2 = e^{\varepsilon}$ and $\hat{w}_1 = e^{100}$, $\hat{w}_2 = e^{\varepsilon}$, then $\|\ln(w) - \ln(\hat{w})\|_\infty$ can be made arbitrarily large by increasing the constant 1000, yet, $\hat{w}$ provides good predictions: $|\frac{w_1}{w_1+w_2} - \frac{\hat{w}_1}{\hat{w}_1+\hat{w}_2}| \leq \varepsilon$. Moreover, also these works analyze non-adaptive algorithms and assume a bound on the ratio of the weights, and therefore suffer our $\Omega(n^2 \log n)$ lower bound.

We also mention a separate line of work that focuses on developing statistical tests to determine whether a given dataset of comparisons is consistent with an MNL model (Seshadri and Ugander, 2019; Makur and Singh, 2025; Rastogi et al., 2022). These works are complementary to ours in that they address model validation rather than estimation, and they do not provide algorithms for learning the underlying MNL parameters.

MNL Ranking/Selection. Other classical problems involving MNLs include: (i) sorting/ranking the weights (Falahatgar et al., 2017, 2018; Chen et al., 2022; Szörényi et al., 2015; Ren et al., 2019), (ii) finding the top-k items with largest weight (Jang et al., 2017; Chen et al., 2017, 2018; Chen and Suh, 2015; Kalyanakrishnan et al., 2012), and (iii) finding the item of maximum weight (Even-Dar et al., 2002; Mannor and Tsitsiklis, 2004). Some works make assumptions about the weights (e.g., adjacent weights are sufficiently separated) and seek an exact output with high probability (Jang et al., 2017), while others do not make further assumptions but only require a probably approximately correct (PAC) output (Szörényi et al., 2015). These problems have also been explored in the "dueling bandits" literature (Bengs et al., 2021).

While we will use an $O(\frac{n \log n}{\varepsilon^2})$ approximate sorting algorithm by Falahatgar et al. (2018) as a first step in our algorithm, these results are not sufficient by themselves to learn an MNL under our definition. Our lower bound will be proved by showing that the top-$\frac{n}{2}$ problem (and the ranking problem) reduces to our MNL learning problem. Interestingly, despite this, we obtain an $O(\frac{n \log n}{\varepsilon^2})$ learning algorithm which is (asymptotically) as efficient as the best possible algorithm for MNL

ranking (Falahatgar et al., 2018).

Distribution Testing with Conditional Samples. Our problem can also be described in the context of conditional sampling. Let μ be a hidden distribution over $[n]$. Algorithms can, adaptively, make the following types of queries: chosen a set $S \subseteq [n]$, an oracle returns an item of S sampled according to distribution μ conditioned on S.[4] The goal in distribution testing is usually to make the smallest number of queries to establish whether μ satisfies certain properties, such as, e.g., uniformity (Canonne, 2020). However, the problem of estimating the probability $\mu(i)$, for $i \in [n]$, has also been considered (Chakraborty et al., 2013; Canonne et al., 2015; Adar et al., 2026).

Our problem, on the other hand, asks for the minimum number of queries to accurately estimate $\mu(i \mid S)$ for each $i \in S \subseteq [n]$.[5] Indeed, the distribution μ can be seen as the weights of an MNL M and therefore, $\mu(i \mid S) = M_S(i)$ for $i \in S \subseteq [n]$. Observe that having an estimate only for $\mu(i)$ is equivalent to an estimate of the winning distribution on the full slate—insufficient to estimate the winning distribution for smaller slates. Some algorithms provide a multiplicative $(1 \pm \varepsilon)$ estimate for $\mu(i)$ for $i \notin B$ where B is a set such that $\sum_{b \in B} \mu(b) \leq \varepsilon$; this is a stronger property than an additive approximation of the full slate. However, it still cannot provide accurate estimates for slates that are either subsets of B or that span across B and $[n] \setminus B$. Note also that it can be $|B| = \Theta(n)$ meaning that the distribution of most slates cannot be estimated. From a technical standpoint, Chakraborty et al. (2013) achieve this guarantee by building a complete binary tree where edges are labeled with probabilities. This idea bears some high level similarities with our estimation-forest, but details differ. Indeed, their tree is static while the topology of our forest is adaptively chosen, which is crucial for a tight $O(n \log n)$ bound. Moreover, their tree is populated by querying slates of arbitrary size, while we only query slates of size two. Finally, as argued above, the guarantees provided by their tree are insufficient to estimate the winning distributions of all slates. Recent work has also focused on different query models (Adar, 2025; Pradhan and Roy, 2025; Meel et al., 2025); however, their results are incomparable to ours.

We remark that a common paradigm for designing distribution testing algorithms in the traditional (unconditional) setting is that of *testing by learning* (see, e.g. Canonne (2022)), in which a property is tested by first approximately learning the underlying distribution, and then checking whether the learned distribution has the property in question. Our work serves as a conditional counterpart to this paradigm that works for the more challenging case in which the property being tested requires approximating the behavior of all conditional distributions.

RUM Learning. MNLs are a special case of RUMs; hence, algorithms for learning RUMs on all slates could be used to learn an MNL. However, the best known algorithms for general RUM learning require exponentially many queries to slates of size $\Theta(\sqrt{n})$ (Chierichetti et al., 2024). In contrast, we show that MNLs can be learned using only $O(n \log n)$ queries to slates of size two.

3 Technical Preliminaries

Let $U = [n] = \{1, \ldots, n\}$ be a universe of items. For a probability distribution P over $[n]$, let $P(i)$ denote the probability of the item $i \in [n]$. For distributions P, Q, let $\|P - Q\|_1 := \sum_{i \in [n]} |P(i) - Q(i)|$ be the ℓ_1-*distance*, which is also twice the total variation distance, and let $\|P - Q\|_\infty := \max_{i \in [n]} |P(i) - Q(i)|$ be the ℓ_∞-*distance*. Let $X \sim \mathrm{Ber}(\mu)$ denote a random variable following a

[4] If S has probability 0, the oracle returns a uniform at random item from S.

[5] In our proofs, we will assume that the weights are strictly positive for simplicity. However, the same algorithms also work when weights of zero are allowed, as we show in Section E.

Bernoulli distribution with mean μ and let $X \sim \mathrm{Bin}(n,p)$ denote a random variable following a binomial distribution with n trials and head probability p. Also let $X \sim \mathrm{Geom}(p)$ denote a random variable following a geometric distribution with parameter $p \in (0,1]$; in particular, $\Pr_{X\sim\mathrm{Geom}(p)}[X = k] = (1-p)^{k-1}p$, for $k \geq 1$. For any $x, y \in \mathbb{R}$, we denote by $x \pm y$ the interval $[x-y, x+y]$ and for any $x \in \mathbb{R}, \varepsilon \in (0,1)$, we denote by $(1\pm\varepsilon)x$ the interval $[(1-\varepsilon)x, (1+\varepsilon)x]$.

Ordered Clusterings and Directed Weightings. An *ordered clustering* of $[n]$ is given by an ordered partition $(C_1, \ldots, C_T)$ of $[n]$ and a corresponding list $(c_1, \ldots, c_T)$ of *centers* such that $c_i \in C_i$ for each i. Here, for $v \in [n]$, let $\gamma(v) \in [T]$ be the unique index such that $v \in C_{\gamma(v)}$; we call $\gamma(v)$ the *cluster index* of v.

Let $F = ([n], E)$ be an undirected forest supported on $[n]$. For $u, v \in [n]$ in the same connected component of F, let $P(u,v)$ be the (unique) path in F from u to v. We use $d(u,v)$ to denote the (unweighted/hop) distance in F between vertices u and v, where if u and v are in different connected components, we define $d(u,v) = \infty$.

Let $\vec{E} := \{(u,v) \in V^2 \mid \{u,v\} \in E\}$. A *directed weighting* of the edges of F is a function $r : \vec{E} \to \mathbb{R}_{>0}$ such that $r(u,v) = 1/r(v,u)$. For a path $P = u_1, \ldots, u_t$ in F define $r(P) = \prod_{i=1}^{t-1} r(u_i, u_{i+1})$, and if $t = 1$, let $r(P) = 1$.

4 Overview of Results and Techniques

In this section, we provide a high-level overview of the results obtained in this paper. Our main focus is in showing how to get a tight dependence of $\Theta(n \log n)$ in the support size with adaptive algorithms. Therefore, we will mostly focus in Section 4.1 on our adaptive algorithm for learning an MNL in d_{mul}-error which we believe is our most intuitive algorithm. We will then describe in Section 4.2 how to obtain a tight dependence in ε for the d_1-distance. In Section 4.3 we describe the ideas to make our algorithms non-adaptive, and finally in Section 4.4 we describe the main ideas to obtain our lower bounds.

4.1 Learning MNLs Adaptively in d_{mul}-Error

Our first result is an algorithm to learn an MNL M by making $O\left(\frac{n}{\varepsilon^3}\log n\right)$ adaptive Sample queries to output the weights of an MNL $\hat{M}$ such that $d_{\mathrm{mul}}(M, \hat{M}) \leq \varepsilon$. Observe that $M_S(i) = \frac{w_i}{\sum_{s\in S} w_s} = \frac{1}{\sum_{s\in S} \frac{w_s}{w_i}}$. Therefore, if we had access to a multiplicative estimate of the ratio w_i/w_j for each pair $i, j \in [n]$, we could provide a good multiplicative estimate for M_S for each slate S. Unfortunately, this has two issues. (i) In general, this ratio can be unbounded and therefore, producing a multiplicative estimate could in principle cost an unbounded number of queries. (ii) If we aim to obtain an algorithm with query complexity $o(n^2)$, we simply cannot afford to query all the pairs.

To circumvent these issues, we instead construct a *sparse* graph on the items of $[n]$ that contains estimates of the ratio w_i/w_j along each edge $\{i,j\}$, and then use this graph to compute $\hat{M}$. At a high level, we produce a forest F such that: (i) if two items are close to each other in F, we can get an estimate of their ratios, (ii) if two items are far away in F, then their ratio is negligible. We will also need some technical properties to ensure that we can obtain a valid MNL $\hat{M}$ from the forest. The following definition formalizes the properties we need.

Definition 4 ((t,ε)-Estimation-Forest)**.** Let $t \in \mathbb{Z}, t \geq 2$ and let $\varepsilon \in (0,1)$. A (t,ε)-*estimation-forest* for an MNL supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$ is a tuple $\mathcal{F} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$,

where $F = ([n], E)$ is an undirected forest, r is a directed weighting on F, and $(C_1, \ldots, C_T), (c_1, \ldots, c_T)$ is an ordered clustering over $[n]$. For any $u, v \in [n]$ such that $\gamma(u) \geq \gamma(v)$:

1. if $d(u, v) \leq t$, then $r(P(u, v)) \in (1 \pm \varepsilon) \cdot \frac{w_u}{w_v}$ and $r(P(v, u)) \in (1 \pm \varepsilon) \cdot \frac{w_v}{w_u}$.
2. If $d(u, v) \in (t, \infty)$, then:

$$\sum_{\substack{s \in [n] \\ \gamma(s) \leq \gamma(v)}} \frac{w_s}{w_u} \leq \varepsilon \quad \text{and} \quad \sum_{\substack{s \in \mathcal{C} \\ \gamma(s) \leq \gamma(v)}} r(P(s, u)) \leq \varepsilon,$$

   where $\mathcal{C}$ is the connected component containing both u and v.
3. if $d(u, v) = \infty$, then:

$$\sum_{\substack{s \in [n] \\ \gamma(s) \leq \gamma(v)}} \frac{w_s}{w_u} \leq \varepsilon.$$

   Also, for any u' (resp. v') in the same connected component of u (resp. v), it holds that $\gamma(u') > \gamma(v')$.
4. if $\gamma(u) = \gamma(v)$, then $d(u, v) \leq t$.

In Section 6.3, we show that we can use a (t, ε)-estimation-forest for an MNL M to obtain an MNL $\hat{M}$ such that $d_{\text{mul}}(M, \hat{M}) \leq O(\varepsilon)$ (Theorem 29).

On Choosing the Estimation-Forest Topology. Interestingly, for the purpose of constructing $\hat{M}$, it turns out that the specific value of t is irrelevant. This observation allows us to reduce the problem of learning M to that of constructing a (t, ε)-estimation-forest for a single, arbitrary choice of t. The central challenge then lies in designing an efficient topology for the estimation-forest.

The most natural topology would be a path on the items (after a noisy-sorting step). However, along a path, two items of comparable weight can be separated by a super-constant distance $d = \omega(1)$. To preserve property 2 of Definition 4, one would then need to construct a (d, ε)-estimation-forest, which would incur a query cost of $\Omega(nd^2 \log n)$. Since d can be as large as $\Theta(n)$, this is clearly suboptimal, suggesting the need for a topology with low diameter. Note that even a complete binary tree also can yield super-constant length paths, implying an $\omega(n \log n)$ query cost.

On the other hand, to achieve a very small diameter, one might consider a star or a tree topology with unbounded arity. However, in these cases, one would need to estimate extremely large weight ratios, leading to high query complexity. This, in fact, explains why a disconnected graph is required.

Another natural direction would be to consider general (non-acyclic) graphs. In fact, we could consider a path with skips to decrease the diameter (perhaps exploiting modern shortcutting results (Kogan and Parter, 2022)). The difficulty is that, in a cyclic graph, the weight of an item depends on the particular path chosen, and different paths can yield inconsistent estimates. Thus, acyclicity of the topology is essential to ensure that an explicit MNL can be extracted from it.

In Section 6.2, we present an efficient algorithm for constructing an $(O(1), \varepsilon)$-estimation-forest. The resulting topology takes the form of a forest of *lobster graphs*: items are first clustered together, as described below, and a forest of unbounded-arity trees is then constructed over the resulting cluster centers. The arity of each tree is not predetermined but is instead adaptively chosen as the algorithm progresses, in order to balance estimation accuracy and query efficiency. Interestingly, the diameter of the trees in our forest can be super-constant. However, each tree will have the property that if two items are at distance more than $O(1)$, then one of the two is so much larger than the other that their ratio can be taken to be infinite without incurring a large error. Thanks to this property, from the perspective of any single item, one can consider the tree to have constant diameter and lose at most $O(\varepsilon)$ in the final estimate.

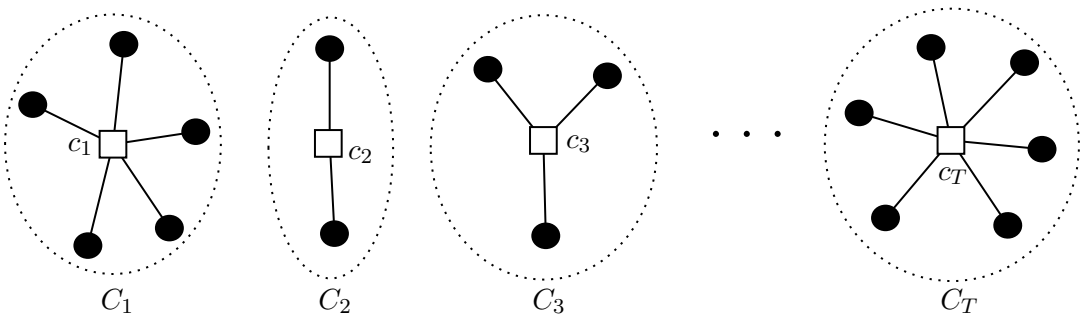


Figure 1: The structure of an (A_1, A_2, ε)-cluster graph. The vertices of the graph are the items $[n]$ of the MNL, the cluster centers are depicted as white-filled squares, while the other items are represented by black circles. Items in the same cluster have similar weight (within a factor of A_1 of each other). Clusters further to the right contain items of higher weights. Associated with each edge $\{u, v\}$, and each direction (say, $u \to v$), is an estimate $r(u, v)$ of the ratio w_u/w_v.

Building the Estimation-Forest. We now go more into the details of our solution to efficiently build an estimation-forest. When constructing the estimation-forest, some ratio estimates might be costlier to obtain than others. In order to maintain a low query complexity, we leverage the fact that if two items have similar weights, fewer queries are required to estimate the ratio of their weights. In the first step to build our estimation-forest, we exploit this observation via a pre-processing step, which sorts the items in approximately increasing order of weights, and produces clusters of similar items resulting in a *cluster graph*, defined as follows.

Definition 5 (Cluster Graph)**.** An (A_1, A_2, ε)-cluster graph for an MNL supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$, is a tuple $\mathcal{G} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$, where $F = ([n], E)$ is an undirected forest, r is a directed weighting on F, and $(C_1, \ldots, C_T), (c_1, \ldots, c_T)$ is an ordered clustering over $[n]$, satisfying:

1. For any $i \in [T]$ and any item u in the cluster C_i we have:
$$\frac{1}{A_1} \leq \frac{w_u}{w_{c_i}} \leq A_1.$$
2. For any $i, j \in [T]$ with $i > j$ we have:
$$\frac{w_{c_i}}{w_{c_j}} \geq A_2.$$
3. E consists of all the edges of the form $\{c_i, u\}$ for all choices of i and of $u \in C_i$. Moreover the weight $r(u, v)$ of any edge $\{u, v\} \in E$ satisfies:
$$r(u, v) \in (1 \pm \varepsilon)\frac{w_u}{w_v} \quad \text{and} \quad r(v, u) = \frac{1}{r(u, v)} \in (1 \pm \varepsilon)\frac{w_v}{w_u}.$$

In Section 6.1, we show how to obtain a cluster graph. Our algorithms employs a noisy sorting procedure of Falahatgar et al. (2018) as a subroutine and builds on it to partition the vertices and compute the edge weights r. We show in Figure 1 a cluster graph produced by our algorithm.

Observe that a cluster graph is not yet an estimation-forest. Indeed, there might be items in different clusters (but close in the ordering) whose ratio is constant. To obtain an estimation-forest, we add extra edges between some pairs of centers. We do so in an iterative way, starting from the center of the last cluster and moving backwards. *A priori*, these multiplicative estimates can

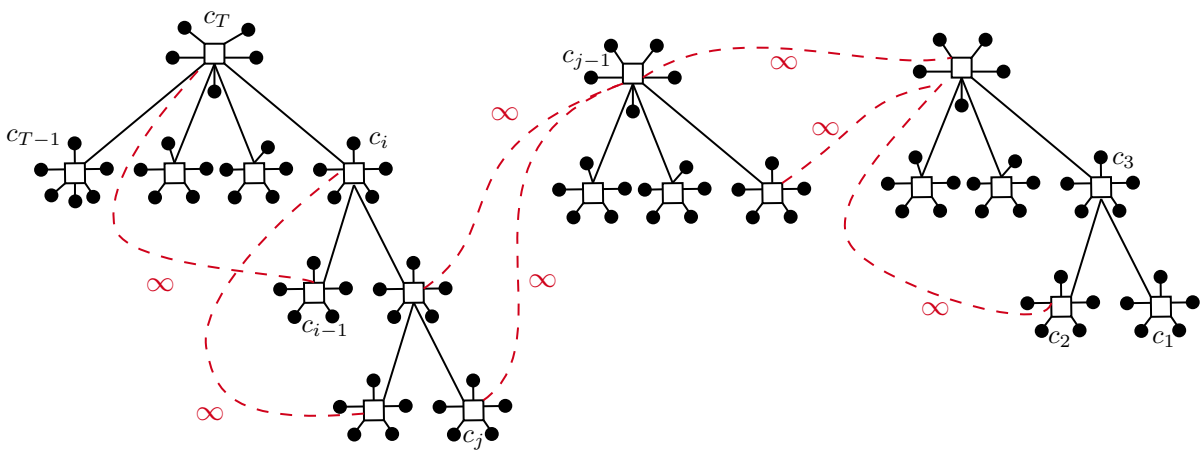


Figure 2: The structure of an estimation-forest constructed by Algorithm 4 in Section 6. White squares represent cluster centers, while black circles represent the other items of $[n]$. A new level in the forest is created when two nodes are compared and the estimate of their ratio is "∞". If this happens twice consecutively (for the parent node and the children with smallest estimated weight), then a new tree is created. In the figure, we have $i < T - 1$ and $j < i - 1$.

potentially be costly to obtain, since the ratio between the weights of distinct cluster centers could be arbitrarily large. In order to maintain a low query complexity, we employ a careful thresholding strategy. This ensures that we only require an accurate estimate of the ratio when this is not too large to make a significant difference in the MNL winning distributions. For instance, if the ratio between two items is greater than $\Omega(\frac{n}{\varepsilon})$, then it is safe to act as if the second item's weight is infinitely larger than the first, as this approximation only causes a d_1-error of magnitude $O(\varepsilon)$. When we find two clusters that are incomparable, we restart the iteration process from the last cluster that was comparable. It can be shown that this leads to an $(O(1), \varepsilon)$-estimation-forest (see Theorem 21). We show in Figure 2 a forest that can be produced by our algorithm.

In summary, our algorithm has *three* phases. In the first phase, we construct a $(\Theta(1), \Theta(1), \Theta(\varepsilon))$-cluster graph. In the second phase, we extend the cluster graph to a $(\Theta(1), \Theta(\varepsilon))$-estimation-forest. Finally, in the third phase, we use the forest to recover an estimate of the MNL weights. A representation of the steps in our algorithm is in Figure 3. The first two phases require at most $O(\frac{n \log n}{\varepsilon^3})$ queries, while the last one does not make any further queries, yielding our main result:

Theorem 6. *Choose any $\varepsilon \in (0, 1)$ and $\delta = n^{-c}$ for a constant $c > 0$. There exists an adaptive randomized algorithm that, with probability at least $1 - \delta$, makes $O\left(\frac{n \log n}{\varepsilon^3}\right)$ Sample queries and solves the MNL Learning Problem for d_{mul}-error on $[n]$ with accuracy parameter ε. Moreover, the algorithm only queries pairs and runs in time proportional to the number of queries.*

4.2 Learning MNLs Adaptively in d_1-Error

In order to obtain a tight dependence in ε for d_1-error guarantees, we relax the notion of estimation-forest to the following:

Definition 7 (ε-weak-estimation-forest)**.** Let $\varepsilon \in (0, 1)$. An *ε-weak-estimation-forest* for an MNL supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$ is a tuple $\mathcal{F} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$, where $F = ([n], E)$ is an undirected forest, r is a directed weighting on F, and $(C_1, \ldots, C_T), (c_1, \ldots, c_T)$ is an ordered clustering over $[n]$. For any $u \in [n]$ the following holds:

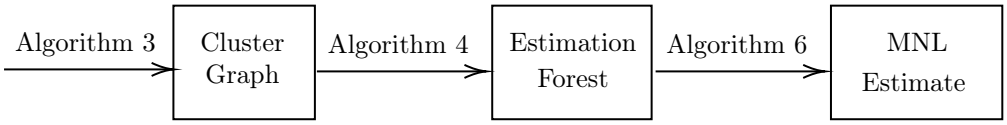


Figure 3: The structure of our algorithm to learn MNLs adaptively. The non-adaptive algorithm follows the same overall structure, but the first two steps are replaced by `QuicksortClustering` (described in Proposition 35) and by Algorithm 11 respectively.

1. let $\mathcal{C}$ be the connected component of u, then for any $A \subseteq \{s \in \mathcal{C} \mid \gamma(s) \leq \gamma(u)\}$ it holds that:
$$\sum_{s \in A} \left| \frac{w_s}{w_u} - r(P(s,u)) \right| \leq \varepsilon \cdot \left(1 + \sum_{s \in A} \frac{w_s}{w_u}\right)$$
2. for any $v \in [n]$ in a different connected component than u and such that $\gamma(v) \leq \gamma(u)$, we have:
$$\sum_{\substack{s \in [n] \\ \gamma(s) \leq \gamma(v)}} \frac{w_s}{w_u} \leq \varepsilon.$$
Also, for any u' (resp. v') in the same connected component of u (resp. v), it holds that $\gamma(u') > \gamma(v')$.

We will show that an ε-weak-estimation-forest is enough to obtain a d_1-error of $O(\varepsilon)$ (Theorem 34), and that such weaker notion of estimation forest can be constructed more efficiently (Theorem 33). Therefore, we obtain the following:

Theorem 8. *Choose any $\varepsilon \in (0,1)$ and $\delta = n^{-c}$ for a constant $c > 0$. There exists an adaptive randomized algorithm that, with probability at least $1 - \delta$, makes $O\left(\frac{n \log n}{\varepsilon^2}\right)$ Sample queries and solves the MNL Learning Problem for d_1-error on $[n]$ with accuracy parameter ε. Moreover, the algorithm only queries pairs and runs in time proportional to the number of queries.*

4.3 Learning MNLs Non-Adaptively

We next present an algorithm to learn MNLs non-adaptively, i.e., by making a single batch of queries. In order to do this we leverage the following reduction.

Lemma 9. *Given an adaptive algorithm for learning MNLs with the* Sample *oracle that queries any pair of items at most m times, one can construct a non-adaptive algorithm for the same problem that makes at most $m\binom{n}{2} = O(mn^2)$ queries.*

Proof. The non-adaptive algorithm queries each pair m times and then simulates the adaptive algorithm by replacing each Sample oracle call with a revealed response from the set of non-adaptive queries. □

For constant ε, the number of Sample queries made to any pair $\{u, v\} \subseteq [n]$ of items by the adaptive algorithms described above could be as high as $\tilde{O}(n)$; this would naively yield an $\tilde{O}(n^3)$-algorithm. Instead, we design an algorithm with query complexity $\tilde{O}(n^2)$. To accomplish this, we modify the adaptive algorithm to obtain a new (adaptive) algorithm that has a worse overall query complexity than the algorithms of Theorem 6 and Theorem 8, but allows us to uniformly bound the number of queries made to each pair of items. In particular, in Section 8 we show the following result.

Theorem 10. *Choose any $\varepsilon, \delta \in (0,1)$. There exists an adaptive randomized algorithm that, with probability at least $1-\delta$, queries each pair at most $O\left(\frac{\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)$ times and solves the MNL Learning Problem for d_{mul}-error on $[n]$ with accuracy parameter ε.*

To obtain this result, we use three new technical ingredients. First, we make use of a different algorithm to approximately order the items of the MNL (Section D). This algorithm, which is a straight-forward adaptation of the classical Quicksort algorithm, makes more queries than the previous one overall, but guarantees a uniform upper bound on the number of queries on each pair of items. Second, we introduce a new algorithm to construct the estimation-forest. This algorithm only needs to make $O(|C_j|\log^2 n)$ comparisons between any pair $\{c_i, c_j\}$ of cluster centers (with $i > j$) whenever the ratio w_{c_i}/w_{c_j} is estimated. Finally, we introduce a subroutine (Algorithm 10) that allows one to amortize the cost of estimating the ratio w_{c_i}/w_{c_j} among all the pairs of the form $\{c_i, s\}$, where s belongs to the cluster C_j. This allows one to distribute the $O(|C_j|\log^2 n)$ cost nearly equally among all items in C_j, and hence to guarantee each pair is queried at most $O(\log^2 n)$ times.

Combining Theorem 10 and Lemma 9 yields:

Corollary 11. *Choose any $\varepsilon, \delta \in (0,1)$. There exist a non-adaptive algorithm that, with probability at least $1-\delta$, makes at most $O\left(\frac{n^2\cdot\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)$ queries and solves the MNL Learning Problem for d_{mul}-error on $[n]$ with accuracy parameter ε.*

4.4 Lower Bounds

We prove lower bounds that show that our adaptive algorithms are asymptotically optimal in both n and ε, and that our non-adaptive algorithm has nearly-optimal (at most a $\log n$ factor away from optimal) dependence on n.

For learning MNLs with adaptive queries to Sample, in Section 9.2 we show the following.

Theorem 12. *Any (possibly randomized and adaptive) algorithm that, given in input $\varepsilon, \delta \in (0, \frac{1}{32})$ and access to a* Sample *oracle for any MNL M, outputs an MNL $\hat{M}$ satisfying:*

$$\Pr[d_\infty(M, \hat{M}) \le \varepsilon] \ge 1-\delta,$$

must make $\Omega(\frac{n}{\varepsilon^2}\log\frac{n}{\delta})$ queries in the worst case.

Note that this lower bound holds even for d_∞, while our matching algorithm works for d_1. For the case of d_{mul}-error we can show a stronger dependency on ε which matches our upper bound for learning in d_{mul}-error:

Theorem 13. *Any (possibly randomized and adaptive) algorithm that, given in input $\varepsilon, \delta \in (0, \frac{1}{128})$ and access to a* Sample *oracle for any MNL M, outputs an MNL $\hat{M}$ satisfying:*

$$\Pr[d_{mul}(M, \hat{M}) \le \varepsilon] \ge 1-\delta,$$

must make $\Omega(\frac{n}{\varepsilon^3}\log\frac{n}{\delta})$ queries in the worst case.

For the non-adaptive case, in Section 9.3 we show the following.

Theorem 14. *Any (possibly randomized) non-adaptive algorithm that, given in input $\varepsilon \in (0, \frac{1}{32})$ and access to a* Sample *oracle for any MNL M, outputs an MNL $\hat{M}$ satisfying:*

$$\Pr[d_\infty(M, \hat{M}) \le \varepsilon] \ge \frac{9}{10},$$

must make $\Omega(\frac{n^2}{\varepsilon^2}\log n)$ queries in the worst case.

The lower bounds we provide are based on reductions from the problem of approximately identifying the $\frac{n}{2}$ coins with the largest probability of *heads* in a set of n biased coins.

5 Algorithmic Primitives

We will make use of the following two key subroutines, `Compare` and `EstimateRatio`, and we will frequently refer to their guarantees provided below.

Algorithm 1 `Compare`$(i, j, c, \varepsilon, \delta)$

Input: Two items i and j of $[n]$, parameters $c, \varepsilon, \delta \in (0, 1)$, and access to a Sample oracle for an MNL M supported on $[n]$.
Output: Estimates $\hat{p}_i$ and $\hat{p}_j$ of $M_{\{i,j\}}(i)$ and $M_{\{i,j\}}(j)$ respectively.
Make $m = \frac{20}{c \cdot \varepsilon^2} \ln\left(\frac{6}{\delta}\right)$ queries to Sample$(\{i, j\})$ and let m_i and m_j be the number of queries that return i and j respectively.
Let $\hat{p}_i = \frac{m_i}{m}$ and $\hat{p}_j = \frac{m_j}{m}$
if $\hat{p}_i < c/2$ **then**
    return $(0, \hat{p}_j)$
if $\hat{p}_j < c/2$ **then**
    return $(\hat{p}_i, 0)$
return $(\hat{p}_i, \hat{p}_j)$

In particular, a simple consequence of standard tail bounds is the following guarantee, which we prove for completeness in Section A.

Lemma 15 (`Compare` guarantees)**.** *For any $c, \varepsilon, \delta \in (0, 1)$, `Compare`$(i, j, c, \varepsilon, \delta)$ makes $O\left(\frac{1}{c\varepsilon^2} \log \frac{1}{\delta}\right)$ queries and outputs a pair $(\hat{p}_i, \hat{p}_j)$ that, with probability at least $1 - \delta$ satisfies, for $k \in \{i, j\}$:*

1. *If $M_{\{i,j\}}(k) \leq c/4$, then $\hat{p}_k = 0$,*
2. *If $M_{\{i,j\}}(k) \geq c$, then $\hat{p}_k \neq 0$,*
3. *If $\hat{p}_k \neq 0$ then $(1 - \varepsilon) M_{\{i,j\}}(k) \leq \hat{p}_k \leq (1 + \varepsilon) M_{\{i,j\}}(k)$.*

This in turn implies the following lemma, which shows guarantees on the behavior of `EstimateRatio`. This is also proved in Section A.

Algorithm 2 `EstimateRatio`$(i, j, \alpha, \varepsilon, \delta)$

Input: Two items $i, j \in [n]$, parameters $\alpha, \varepsilon, \delta \in (0, 1)$, and access to a Sample oracle for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$.
Output: An estimate $r(i, j)$ of the ratio of the weights w_i / w_j in the MNL.
Let $c = \alpha/(\alpha + 1)$.
$(\hat{p}_i, \hat{p}_j) =$ `Compare`$(i, j, c, \varepsilon/3, \delta)$
if $\hat{p}_i = 0$ **then**
    return $r(i, j) = 0$
if $\hat{p}_j = 0$ **then**
    return $r(i, j) = \infty$
return $r(i, j) = \frac{\hat{p}_i}{\hat{p}_j}$

Lemma 16 (`EstimateRatio` Guarantees)**.** *Given two items i and j of $[n]$, and parameters α, ε, and δ in $(0, \frac{1}{2}]$, the algorithm* `EstimateRatio`$(i, j, \alpha, \varepsilon, \delta)$ *makes $O(\frac{1}{\alpha\varepsilon^2}\log\frac{1}{\delta})$ queries and produces an estimate $r(i,j)$ of the ratio $\frac{w_i}{w_j}$ that, with probability $1-\delta$, satisfies the following guarantees:*

1. *If $\frac{w_i}{w_j} \le \frac{\alpha}{3\alpha+4}$, then $r(i,j) = 0$.*
2. *If $\frac{w_i}{w_j} \ge \frac{3\alpha+4}{\alpha}$, then $r(i,j) = \infty$.*
3. *If $\frac{w_i}{w_j} \le \frac{1}{\alpha}$, then $r(i,j) \neq \infty$, and if $\frac{w_i}{w_j} \ge \alpha$ then $r(i,j) \neq 0$.*
4. *Whenever $r(i,j) \notin \{0, \infty\}$:*

$$r(i,j) \in (1 \pm \varepsilon)\frac{w_i}{w_j} \quad \text{and} \quad \frac{1}{r(i,j)} \in (1 \pm \varepsilon)\frac{w_j}{w_i}.$$

6 Adaptive Algorithm for d_{mul} Error

In this section we describe our adaptive algorithm for d_{mul}-error. The algorithm comprises three phases: (i) construct a cluster graph to approximately sort and group together items of similar weights (Section 6.1), (ii) extend the cluster graph to an estimation-forest (Section 6.2), and (iii) extract an MNL from the estimation-forest (Section 6.3).

6.1 Constructing a Cluster Graph with $O\left(\frac{n\log n}{\varepsilon^2}\right)$ Queries

In this section, we describe and analyze an algorithm to construct a cluster graph as defined in Definition 5. This makes up the first phase of our adaptive algorithm to learn MNLs.

The first ingredient to obtain this result is an adaptive algorithm developed by Falahatgar et al. (2018) to sort items in approximately increasing order of weight by querying a noisy pairwise comparison oracle. We consider the following definition.

Definition 17 (ε_o-ordering)**.** An *ε_o-ordering* for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$ is an ordering $(s_1, \ldots, s_n)$ of the items of $[n]$ such that, for any pair i, j with $i < j$: $(1-\varepsilon_o)w_{s_i} \le w_{s_j}$.

The following is a consequence of (Falahatgar et al., 2018, Theorem 9), where we boosted the success probability and made the runtime explicit (proof in Section B provided for completeness).

Theorem 18. *Let $\varepsilon_o, \delta \in (0,1)$. There is an algorithm that given access to a* Sample *oracle for an MNL M supported on $[n]$, with probability at least $1-\delta$, makes $O\left(\frac{n\log(n/\delta)}{\varepsilon_o^2}\cdot\left(1+\frac{\log(1/\delta)}{\log n}\right)\right)$ queries and returns an ε_o-ordering of the items of M. Moreover, all the queries made by the algorithm are to slates of size two and the algorithm runs in time proportional to the number of queries.*

Below, we introduce the main algorithm of this section: `ClusterSort` (Algorithm 3). At a high level, the algorithm first computes an $O(1)$-ordering of the items as described above, and then it partitions them into clusters that are adjacent in this ordering. The center of a cluster is always chosen to be the first item in the cluster to appear in the $O(1)$-ordering. At each iteration the algorithm tries to add the ℓth item s_ℓ in the ordering to the cluster centered at some item c_i. If the ratio w_{s_ℓ}/w_{c_i} is estimated to be too large, the algorithm simply starts a new cluster.

We begin by analyzing the algorithm's running time and query complexity.

Proposition 19 (Complexity of `ClusterSort`)**.** *With probability at least $1-\delta$,* `ClusterSort`$(\alpha, \varepsilon, \delta)$ *makes $O\left(n\cdot\log(\frac{n}{\delta})\cdot\left(\frac{1}{\alpha\varepsilon^2}+\frac{\log(1/\delta)}{\log n}\right)\right)$ queries to the* Sample *oracle.*

Algorithm 3 `ClusterSort`$(\alpha, \varepsilon, \delta)$

Input: Access to a Sample oracle for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$, parameters $\varepsilon \in (0, 1/7)$, $\alpha, \delta \in (0, 1)$.
Output: A $(\frac{2}{\alpha}, \frac{1}{\alpha}, \varepsilon)$-cluster graph $\mathcal{G} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$.
$\tau = \frac{3(1+\varepsilon)}{2\alpha}$.
$\mathit{ClusterList} = \varnothing$, $\mathit{Centers} = \varnothing$, $E = \varnothing$
Construct an $\frac{1}{3}$-ordering $S = (s_1, \ldots, s_n)$ for M using the algorithm of Theorem 18 with error probability $\delta/2$.
$i = 1$, $j = 1$, $\ell = 2$
$c_i = s_1$
while $\ell \leq n$ **do**
    $r(s_\ell, c_i) =$ `EstimateRatio`$(s_\ell, c_i, \frac{2\alpha}{3}, \varepsilon, \frac{\delta}{2n})$
    if $r(s_\ell, c_i) > \tau$ **then**
        $C_i = \{s_j, \ldots, s_{\ell-1}\}$
        $\mathit{ClusterList} = \mathit{ClusterList} \circ (C_i)$, $\mathit{Centers} = \mathit{Centers} \circ (c_i)$
        Add to E edges $\{s_a, c_i\}$ for $a \in \{j+1, \ldots, \ell-1\}$ with weight $r(s_a, c_i)$
        $i = i + 1$, $c_i = s_\ell$, $j = \ell$
    $\ell = \ell + 1$
$C_i = \{s_j, \ldots, s_n\}$
$\mathit{ClusterList} = \mathit{ClusterList} \circ (C_i)$, $\mathit{Centers} = \mathit{Centers} \circ (c_i)$
Add to E edges $\{s_a, s_j\}$ for $a \in \{j+1, \ldots, n\}$ with weight $r(s_a, s_j)$
return $(F = ([n], E), r, \mathit{ClusterList}, \mathit{Centers})$

Proof. The first part of the algorithm (the $\frac{1}{3}$-ordering) makes $O\left(n \log(n/\delta) \cdot \left(1 + \frac{\log(1/\delta)}{\log n}\right)\right)$ queries by Theorem 18. The while loop is executed $O(n)$ times. At each execution we call `EstimateRatio` with parameters $2\alpha/3, \varepsilon$, and δ/n and hence the number of Sample queries per call is $O\left(\frac{1}{\alpha\varepsilon^2} \log \frac{n}{\delta}\right)$ by Lemma 16. Hence, the result follows. □

We then show that the algorithm correctly computes the cluster graph.

Theorem 20 (Guarantees for `ClusterSort`)**.** *Let $\alpha, \delta \in (0, 1)$, and $\varepsilon \in (0, 1/7)$. Let $\mathcal{G}$ be the output of* `ClusterSort`$(\alpha, \varepsilon, \delta)$*. Then, with probability at least $1 - \delta$, $\mathcal{G}$ is a $(\frac{2}{\alpha}, \frac{1}{\alpha}, \varepsilon)$-cluster graph.*

Proof. Let $\mathcal{G} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be the output of `ClusterSort`$(\alpha, \varepsilon, \delta)$. We note that the $\frac{1}{3}$-ordering procedure succeeds with probability at least $1 - \frac{\delta}{2}$, and that each call to `EstimateRatio` succeeds with probability at least $1 - \frac{\delta}{2n}$. By a union bound they all succeed with probability at least $1 - \delta$. For the rest of the proof, we will assume this holds.

We start by proving the first property of being a cluster graph (Definition 5). When the cluster C_i is created, its center c_i is chosen to be the item in C_i that comes first in the $\frac{1}{3}$-ordering. In particular, for any $s_\ell \in C_i$ we have that:

$$\frac{w_{s_\ell}}{w_{c_i}} \geq \frac{2}{3} \geq \frac{\alpha}{2}. \tag{1}$$

On the other hand, for each item $s_\ell \in C_i$, the algorithm computes the estimate $r(s_\ell, c_i)$ of $\frac{w_{s_\ell}}{w_{c_i}}$ via `EstimateRatio` and finds that $r(s_\ell, c_i) \leq \tau$ (otherwise ℓ would have been placed in a different cluster). Since $r(s_\ell, c_i) \leq \tau$, there are two possibilities: either $r(s_\ell, c_i) = 0$ or $r(s_\ell, c_i) \in (0, \tau]$. We consider the cases separately (and show that the first case cannot happen).

Case 1. Suppose $r(s_\ell, c_i) = 0$, then by the guarantees of `EstimateRatio` it must have been the case that $\frac{w_{s_\ell}}{w_{c_i}} < \frac{2\alpha}{3}$. But the $\frac{1}{3}$-ordering guarantees imply $\frac{w_{s_\ell}}{w_{c_i}} \geq \frac{2}{3}$, giving a contradiction.

Case 2. On the other hand, if $r(s_\ell, c_i) \in (0, \tau]$ then `EstimateRatio` must have returned an accurate estimate for $\frac{w_{s_\ell}}{w_{c_i}}$ and in particular we have:

$$\frac{w_{s_\ell}}{w_{c_i}} \leq \frac{r(s_\ell, c_i)}{1-\varepsilon} \leq \frac{\tau}{1-\varepsilon} \leq \frac{2}{\alpha},$$

since $\varepsilon \leq \frac{1}{7}$, concluding the proof of the first property.

We now prove the second property of Definition 5. Fix a choice of $i \in [T-1]$. Let $\ell^* > i$ be the smallest index such that $r(s_{\ell^*}, c_i) > \tau$. Note that, since there is at least one cluster following C_i, this choice ℓ^* must exist and we must have $\ell^* \leq n$. By construction, we have:

$$C_{i+1} \cup \cdots \cup C_T = \{s_{\ell^*}, \ldots, s_n\}.$$

For all $\ell \geq \ell^*$, we have, by the definition of $\frac{1}{3}$-ordering:

$$\frac{w_{s_\ell}}{w_{s_{\ell^*}}} \geq 1 - \frac{1}{3} = \frac{2}{3}. \tag{2}$$

We now have two possibilities: either $r(s_{\ell^*}, c_i) = \infty$ or $r(s_{\ell^*}, c_i) \in (0, \infty)$. Note that since $r(s_{\ell^*}, c_i) > \tau$, we have that $r(s_{\ell^*}, c_i)$ is non-zero. We consider the two cases separately.

Case 1: If $r(s_{\ell^*}, c_i) = \infty$ then due to the `EstimateRatio` guarantees, we must have:

$$\frac{w_{s_{\ell^*}}}{w_{c_i}} > \frac{3}{2\alpha}, \tag{3}$$

and hence:

$$\frac{w_{s_\ell}}{w_{c_i}} = \frac{w_{s_\ell}}{w_{s_{\ell^*}}} \cdot \frac{w_{s_{\ell^*}}}{w_{c_i}} \overset{(2),(3)}{>} \frac{1}{\alpha}.$$

Case 2: If $r(s_{\ell^*}, c_i) \in (0, \infty)$ then the guarantees of `EstimateRatio` imply that $r(s_{\ell^*}, c_i)$ must be a good approximation to $\frac{w_{s_{\ell^*}}}{w_{c_i}}$. We then have:

$$\frac{w_{s_\ell}}{w_{c_i}} = \frac{w_{s_\ell}}{w_{s_{\ell^*}}} \cdot \frac{w_{s_{\ell^*}}}{w_{c_i}} \overset{(2)}{\geq} \frac{2}{3} \cdot \frac{w_{s_{\ell^*}}}{w_{c_i}} \geq \frac{2}{3(1+\varepsilon)} \cdot r(s_{\ell^*}, c_i) > \frac{2}{3(1+\varepsilon)}\tau = \frac{1}{\alpha}.$$

This yields the second property in Definition 5.

Finally, we prove the third property. For this, we simply argue that the estimates $r(s_\ell, c_i)$ produced by the algorithm are never 0 or ∞ for any item s_ℓ that is part of cluster C_i; the result will then follow from the guarantees of `EstimateRatio`. Note that if an item s_ℓ is placed in the same cluster as c_i then $r(s_\ell, c_i) \leq \tau < \infty$ and hence $r(s_\ell, c_i) \neq \infty$. On the other hand, as we previously argued, by Equation (1) and properties of `EstimateRatio` it also holds $r(s_\ell, c_i) \neq 0$. □

6.2 Constructing an Estimation-Forest with $O\left(\frac{n \log n}{\varepsilon^3}\right)$ Queries

We now present Algorithm 4 to construct an $(O(1), \varepsilon)$-estimation-forest. The algorithm starts by obtaining a $(\frac{2}{\alpha}, \frac{1}{\alpha}, \Theta(\varepsilon))$-cluster graph with ordered clusters $(C_1, \ldots, C_T)$ and their corresponding centers $(c_1, \ldots, c_T)$. Then, starting from c_T the algorithm attempts to estimate the ratios w_{c_i}/w_{c_j} for pairs $\{c_i, c_j\}$ of cluster centers in order to construct an estimation-forest from the cluster graph. However, if this ratio is very large, estimating it accurately will require too many queries. Luckily, in this case, we can pretend as if one center is infinitely heavier (in terms of its MNL weight) than the other; this will contribute to only a small error in the estimated Sample distributions. Intuitively, if c_T wins with probability at least $1 - \varepsilon$ in the slate $\{c_T\} \cup C_j \cup \cdots \cup C_1$, then it is not worth estimating the ratio between w_{c_T} and w_{c_j}. Instead, we can just conclude that the ratio is very large and continue.

In order to get a low query complexity, and guarantee an accurate MNL estimate, it is crucial to design a good thresholding condition to establish when the weight of a cluster center is very large compared to another one for their ratio to be estimated accurately. A threshold too large would force the algorithm to estimate very large ratios, hence have high query complexity. On the other hand, a threshold too small would cause a large error in the estimated Sample distributions.

For each center c_j, we define a potential Z_j that keeps into account both the sizes of the clusters $C_j, \ldots, C_1$ and also their distance in the ordering: $Z_j = \sum_{i=1}^{j} \alpha^{j-i}|C_i|$. We will show that we can deem c_T too large compared to c_j if $\frac{w_{c_j}}{w_{c_T}} \leq \beta_j$ where $\beta_j = \Theta\left(\frac{\varepsilon}{Z_j}\right)$. Even if c_T is too large compared to c_j, it might still be that c_{j+1} is comparable with c_j; therefore, we continue the process with c_{j+1} instead of c_T. If c_{j+1} is also deemed too large, then we start building a new tree starting from c_j.

Algorithm 4 `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$

Input: Parameters $\alpha, \varepsilon, \delta \in (0, 1)$, and access to a Sample oracle for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$.
Output: A $(5, \varepsilon)$-estimation-forest $\mathcal{F} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$.
$\varepsilon_1 = \frac{\varepsilon}{10}$
$(F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T)) =$ `ClusterSort`$(\alpha, \varepsilon_1, \frac{\delta}{3})$
$Z_0 = 0$
for $i = 1, \ldots, T$ **do**
    $Z_i = \alpha \cdot Z_{i-1} + |C_i|$
    $\beta_i = \frac{\alpha^2 \cdot \varepsilon}{8 \cdot Z_i}$
$i = T$
$j = T - 1$
while $j > 0$ **do**
    $r(c_i, c_j) = \max\left\{\texttt{EstimateRatio}(c_i, c_j, \beta_j, \varepsilon_1, \frac{\delta}{6n}), \frac{1}{\alpha^{i-j}}\right\}$
    if $r(c_i, c_j) \neq \infty$ **then**
        $E = E \cup \{\{c_i, c_j\}\}$
        $j = j - 1$
    else if $i = j + 1$ **then**
        $i = j$
        $j = j - 1$
    else
        $i = j + 1$
return $(F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$

We now analyze Algorithm 4. (With a slight abuse of notation, the directed weighting r produced by the algorithm is defined also for some pairs that are not in the set of edges.) Our goal for this section is to prove the following result.

Theorem 21. *Let $\varepsilon, \alpha, \delta \in (0,1)$, then, with probability at least $1-\delta$,* `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$ *makes $O\left(n \log(\frac{n}{\delta}) \cdot \left(\frac{1}{(1-\alpha)\alpha^2\varepsilon^3} + \frac{\log(1/\delta)}{\log n}\right)\right)$* Sample *queries and returns a $(5,\varepsilon)$-estimation-forest.*

We start by bounding the query complexity.

Lemma 22. *Let $\varepsilon, \alpha, \delta \in (0,1)$, then, with probability at least $1-\delta/3$,* `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$ *makes $O\left(n \log(\frac{n}{\delta}) \cdot \left(\frac{1}{(1-\alpha)\alpha^2\varepsilon^3} + \frac{\log(1/\delta)}{\log n}\right)\right)$* Sample *queries.*

Proof. By Proposition 19, `ClusterSort`$(\alpha, \varepsilon_1, \delta/3)$ makes $O\left(n \log(\frac{n}{\delta}) \cdot \left(\frac{1}{\alpha\varepsilon^2} + \frac{\log(1/\delta)}{\log n}\right)\right)$ queries with probability at least $1 - \delta/3$. Observe that for each $j \in [T]$ there are at most two i's for which we make a call to `EstimateRatio`$(c_i, c_j, \beta_j, \varepsilon_1, \frac{\delta}{4n})$. By Lemma 16, each call to `EstimateRatio` costs:

$$O\left(\frac{1}{\beta_j \cdot \varepsilon^2} \cdot \log\left(\frac{n}{\delta}\right)\right) = O\left(\frac{Z_j}{\alpha^2\varepsilon^3} \cdot \log\left(\frac{n}{\delta}\right)\right).$$

Moreover,

$$\sum_{i=1}^{T} Z_i = \sum_{i=1}^{T}\sum_{j=1}^{i} |C_j|\alpha^{i-j} \leq \sum_{i=1}^{T} |C_i| \left(\sum_{j=0}^{T} \alpha^j\right) \leq \sum_{i=1}^{T} |C_i| \left(\sum_{j=0}^{\infty} \alpha^j\right) = \sum_{i=1}^{T} \frac{|C_i|}{1-\alpha} = \frac{n}{1-\alpha}, \quad (4)$$

where the first inequality follows by the fact that each C_i gets summed up at most T times and each time it is multiplied by a different value in $\{\alpha^0, \alpha^1, \ldots, \alpha^T\}$. Thus, the total number of queries made by the algorithm after `ClusterSort` is

$$\sum_{i=1}^{T} 2 \cdot O\left(\frac{Z_i \cdot \log\frac{n}{\delta}}{\alpha^2\varepsilon^3}\right) = O\left(\frac{\log\frac{n}{\delta}}{\alpha^2\varepsilon^3} \cdot \sum_{i=1}^{T} Z_i\right) \leq O\left(\frac{n \log\frac{n}{\delta}}{(1-\alpha)\alpha^2\varepsilon^3}\right). \qquad \square$$

We now move on to proving that the algorithm returns a $(5,\varepsilon)$-estimation-forest. To do this, we need to show that the forest respects the four properties of Definition 4. We will prove this in a series of lemmas. First, we show that items with close cluster indices end up in the same connected component. Recall that given an ordered partition $C_1, \ldots, C_T$ of $[n]$, $\gamma(i)$ for $i \in [n]$ is the unique value $j \in [T]$ such that $i \in C_j$. We have the following result, which entails the fourth and part of the third property of Definition 4.

Lemma 23. *For $\varepsilon, \alpha, \delta \in (0,1)$, let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be the output of* `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$. *Then, if $u, v \in [n]$ are in different connected components and $\gamma(u) > \gamma(v)$, we have that, for any u' (resp. v') in the same connected component of u (resp. v), it holds that $\gamma(u') > \gamma(v')$. Moreover, if $\gamma(u) = \gamma(v)$, then $d(u,v) \leq 2$,*

Proof. Let $\mathcal{C}$ be a connected component of F. Let ψ (resp. ϕ) be the maximum (resp. minimum) index such that $c_\psi \in \mathcal{C}$ (resp. $c_\phi \in \mathcal{C}$). By construction of F, we have $\mathcal{C} = \bigcup_{i=\phi}^{\psi} C_i$. This implies the first part of the lemma. Moreover, since the edges output by `ClusterSort` form a star on each cluster C_i, for any pair of vertices u and v such that $\gamma(u) = \gamma(v)$, we have $d(u,v) \leq 2$. $\square$

We now prove that short paths produce good estimates of the weight ratios; this establishes the first requirement in Definition 4.

Lemma 24. *For $\varepsilon, \alpha, \delta \in (0,1)$, set $\varepsilon_1 = \frac{\varepsilon}{10}$ and let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be the output of* `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$*, and suppose that each call to* `EstimateRatio` *as well as the call to* `ClusterSort` *is successful. Then, for any integer $t \geq 1$ and any $u, v \in [n]$ such that $d(u,v) \leq t$,*

$$(1-\varepsilon_1)^t \cdot \frac{w_u}{w_v} \leq r(P(u,v)) \leq (1+\varepsilon_1)^t \cdot \frac{w_u}{w_v}.$$

In particular, if $d(u,v) \leq 5$, $r(P(u,v)) \in (1 \pm \varepsilon) \cdot \frac{w_u}{w_v}$.

Proof. By Theorem 20, `ClusterSort`$(\alpha, \varepsilon_1, \delta/2)$ returns a $(\frac{2}{\alpha}, \frac{1}{\alpha}, \varepsilon_1)$-cluster graph. Consider $\{c_i, c_j\} \in E$ with $i > j$. By Definition 5, it holds that $w_{c_i} \geq w_{c_j}$. Moreover, since $\{c_i, c_j\} \in E$, $r(c_i, c_j) \neq \infty$, hence by the guarantees of `EstimateRatio` (Lemma 16) we must have that the value $\rho :=$ `EstimateRatio`$(c_i, c_j, \beta_j, \varepsilon_1, \frac{\delta}{4n}) \neq 0$ and specifically:

$$\rho \in (1 \pm \varepsilon_1)\frac{w_{c_i}}{w_{c_j}} \quad \text{and} \quad \frac{1}{\rho} \in (1 \pm \varepsilon_1)\frac{w_{c_j}}{w_{c_i}}. \tag{5}$$

Moreover, by the guarantees of a $(\frac{2}{\alpha}, \frac{1}{\alpha}, \varepsilon_1)$-cluster graph,

$$\frac{w_{c_i}}{w_{c_j}} = \frac{w_{c_i}}{w_{c_{i-1}}} \cdot \frac{w_{c_{i-1}}}{w_{c_{i-2}}} \cdots \frac{w_{c_{j+1}}}{w_{c_j}} \geq \frac{1}{\alpha^{i-j}}. \tag{6}$$

Thus,

$$r(c_i, c_j) = \max\left\{\rho, \frac{1}{\alpha^{i-j}}\right\} \overset{(5),(6)}{\leq} \max\left\{(1+\varepsilon_1)\frac{w_{c_i}}{w_{c_j}}, \frac{w_{c_i}}{w_{c_j}}\right\} \leq (1+\varepsilon_1)\frac{w_{c_i}}{w_{c_j}},$$

and clearly $r(c_i, c_j) \geq \rho \geq (1-\varepsilon_1)\frac{w_{c_i}}{w_{c_j}}$. Similarly,

$$r(c_j, c_i) = \frac{1}{\max\{\rho, \frac{1}{\alpha^{i-j}}\}} = \min\left\{\frac{1}{\rho}, \alpha^{i-j}\right\} \overset{(5),(6)}{\geq} (1-\varepsilon_1)\frac{w_{c_j}}{w_{c_i}},$$

and clearly $r(c_j, c_i) \leq \frac{1}{\rho} \leq (1+\varepsilon_1)\frac{w_{c_j}}{w_{c_i}}$. Note that for each edge $\{u, v\} \in E$, where $\{u, v\} \not\subseteq \{c_1, \ldots, c_T\}$, the value of $r(u, v)$ was computed during the construction of the cluster graph. Therefore $r(u, v) \in (1 \pm \varepsilon_1)\frac{w_u}{w_v}$ and $r(v, u) \in (1 \pm \varepsilon_1)\frac{w_v}{w_u}$ by the definition of cluster graph. Hence these guarantees hold for every pair $\{u, v\} \in E$.

Consider now any $u, v \in [n]$ such that $d(u, v) \leq t$. Let $P(u, v) = a_1, \ldots, a_{t+1}$. We have,

$$r(P(u,v)) = \prod_{i=1}^{t} r(a_i, a_{i+1}) \leq (1+\varepsilon_1)^t \cdot \prod_{i=1}^{t} \frac{w_{a_i}}{w_{a_{i+1}}} = (1+\varepsilon_1)^t \cdot \frac{w_{a_1}}{w_{a_{t+1}}} = (1+\varepsilon_1)^t \cdot \frac{w_u}{w_v}.$$

Note that for $t \leq 5$, $2t \cdot \varepsilon_1 \in (0,1)$, thus, by using that $(1+a)^b \leq 1 + 2ab$ for $a \in [0,1], b \geq 0, 2ab \in (0,1)$, we obtain:

$$r(P(u,v)) \leq (1 + 2 \cdot t \cdot \varepsilon_1)\frac{w_u}{w_v} \leq (1 + 10 \cdot \varepsilon_1)\frac{w_u}{w_v} \leq (1+\varepsilon)\frac{w_u}{w_v}.$$

Similarly,

$$r(P(u,v)) = \prod_{i=1}^{t} r(a_i, a_{i+1}) \geq (1-\varepsilon_1)^t \cdot \frac{w_u}{w_v} \geq (1 - 2 \cdot t \cdot \varepsilon_1)\frac{w_u}{w_v} \geq (1-\varepsilon)\frac{w_u}{w_v},$$

where the last two inequalities hold for $t \leq 5$. In particular, we used that $(1-a)^b \geq 1 - 2ab$, for all $a \in [0,1]$, $b \geq 1$, $2ab \in (0,1)$. □

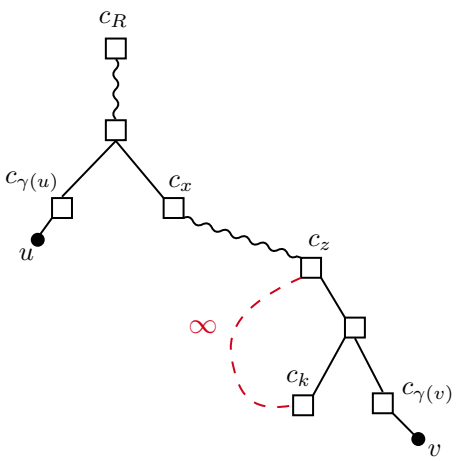


Figure 4: Highlight of some vertices of the estimation-forest computed by Algorithm 4 that are used in the proofs of Lemma 25 and Lemma 27. White squares are cluster centers and black dots are items that are not cluster centers. Curly edges represent paths of length ≥ 0. The dashed edge is not present in the tree but indicates that Algorithm 4 observed $r(c_z, c_k) = \infty$. Vertex c_x is referenced only in the proof of Lemma 27. Note that in the figure we assume that u and v are not centers, but it might also be $u = c_{\gamma(u)}$ or $v = c_{\gamma(v)}$. Moreover, we assume that $c_{\gamma(u)}$ and c_x are siblings but it might also be $c_{\gamma(u)} = c_x$ (similarly for c_k and $c_{\gamma(v)}$). In particular, it holds that $R \geq \gamma(u) \geq x \geq z > k \geq \gamma(v)$, and also $\gamma(u) > z$.

We now prove that items far away in the forest have negligible ratios, concluding the proof of the third point and part of the second point of Definition 4.

Lemma 25. *For $\varepsilon, \alpha, \delta \in (0,1)$, let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be the output of* `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$*, and suppose that each call to* `EstimateRatio` *as well as the call to* `ClusterSort` *is successful. Then, for any $u, v \in [n]$ such that $d(u,v) > 5$ and $\gamma(u) > \gamma(v)$, it holds that $\sum_{s \in H_v} \frac{w_s}{w_u} \leq \varepsilon$, where $H_v = \{s \in [n] \mid \gamma(s) \leq \gamma(v)\}$.*

Proof. Since $d(u,v) \geq 6$, it must be the case that $d(c_{\gamma(u)}, c_{\gamma(v)}) \geq 4$. Consider first the case where u and v are in the same tree. Let R be the largest number in $[T]$ such that c_R is in the same connected component as u and v. We root the tree so that c_R is its root (see Figure 4 for reference). Let c_z be the ancestor of $c_{\gamma(v)}$ at distance 2 from $c_{\gamma(v)}$ in F. Note that c_z exists and is on a level of the tree no smaller than the level of $c_{\gamma(u)}$ since $d(c_{\gamma(u)}, c_{\gamma(v)}) \geq 4$. Therefore, by construction of the tree, it must also hold $\gamma(u) > z$ and by the definition of $(\frac{2}{\alpha}, \frac{1}{\alpha}, \varepsilon)$-cluster graph,

$$w_{c_{\gamma(u)}} \geq w_{c_z}. \tag{7}$$

Observe that, during the construction of this tree, c_z must have been compared with a sibling of $c_{\gamma(v)}$ (or with $c_{\gamma(v)}$ itself) and the result of the estimation must have been ∞, which caused the creation of a new level in the tree. Formally, there must exists c_k, with $z > k \geq \gamma(v)$, such that $r(c_z, c_k) = \infty$. In particular, the result of `EstimateRatio`$(c_z, c_k, \beta_k, \varepsilon_1, \frac{\delta}{4n})$ must have returned ∞. Thus, by Lemma 16,

$$w_{c_z} \geq \frac{w_{c_k}}{\beta_k}. \tag{8}$$

Let $H_{c_k} = \bigcup_{\ell=1}^{k} C_\ell$. Note that $H_v = \bigcup_{\ell=1}^{\gamma(v)} C_\ell \subseteq H_{c_k}$. Consider any $s \in H_{c_k}$. By the definition of

$(\frac{2}{\alpha}, \frac{1}{\alpha}, \varepsilon)$-cluster graph, we have

$$w_u \geq \frac{\alpha}{2} \cdot w_{c_{\gamma(u)}} \overset{(7)}{\geq} \frac{\alpha}{2} \cdot w_{c_z} \overset{(8)}{\geq} \frac{\alpha}{2 \cdot \beta_k} \cdot w_{c_k} \geq \frac{\alpha}{2\beta_k} \cdot \frac{1}{\alpha^{k-\gamma(s)}} \cdot w_{c_{\gamma(s)}}$$
$$\geq \frac{\alpha^2}{4 \cdot \beta_k \cdot \alpha^{k-\gamma(s)}} \cdot w_s \geq \frac{Z_k}{\varepsilon \cdot \alpha^{k-\gamma(s)}} \cdot w_s, \tag{9}$$

where the last inequality is by the definition of β_k. Thus,

$$\sum_{s \in H_{c_k}} \frac{w_s}{w_u} \overset{(9)}{\leq} \sum_{s \in H_{c_k}} \frac{\alpha^{k-\gamma(s)} \cdot \varepsilon}{Z_k} = \sum_{\ell=1}^{k} |C_\ell| \cdot \frac{\alpha^{k-\ell} \cdot \varepsilon}{Z_k} = \frac{\varepsilon}{Z_k} \sum_{\ell=1}^{k} |C_\ell| \cdot \alpha^{k-\ell} = \frac{\varepsilon}{Z_k} \cdot Z_k = \varepsilon.$$

Therefore, by $H_v \subseteq H_{c_k}$, $\sum_{s \in H_v} \frac{w_s}{w_u} \leq \varepsilon$. This concludes the proof for the case where u and v are in the same tree.

Consider now the case where u and v are in different connected components of F. Let z be the smallest integer such that c_z is in the same connected component as u. Then, since $\gamma(u) \geq z$, we have $w_{c_{\gamma(u)}} \geq w_{c_z}$. Note also that it must be $z > \gamma(v)$. Moreover, by construction, $r(c_z, c_{z-1}) = \infty$, otherwise c_{z-1} would be in the same connected component as u. Thus, by the guarantees on `EstimateRatio` in Lemma 16, $w_{c_z} \geq \frac{1}{\beta_{z-1}} \cdot w_{c_{z-1}}$. Let $H_{c_{z-1}} = \bigcup_{\ell=1}^{z-1} C_\ell$, and note that $H_v \subseteq H_{c_{z-1}}$ given that $z - 1 \geq \gamma(v)$. Let $s \in H_{c_{z-1}}$. Similarly to the computation of Equation (9):

$$w_u \geq \frac{\alpha}{2} \cdot w_{c_{\gamma(u)}} \geq \frac{\alpha}{2} \cdot w_{c_z} \geq \frac{\alpha}{2 \cdot \beta_{z-1}} \cdot w_{c_{z-1}} \geq \frac{\alpha}{2 \cdot \beta_{z-1} \cdot \alpha^{z-1-\gamma(s)}} \cdot w_{c_{\gamma(s)}}$$
$$\geq \frac{\alpha^2}{4 \cdot \beta_{z-1} \cdot \alpha^{z-1-\gamma(s)}} \cdot w_s \geq \frac{Z_{z-1}}{\varepsilon \cdot \alpha^{z-1-\gamma(s)}} \cdot w_s.$$

Therefore, similar to previous calculations,

$$\sum_{s \in H_v} \frac{w_s}{w_u} \leq \sum_{s \in H_{c_{z-1}}} \frac{w_s}{w_u} \leq \frac{\varepsilon}{Z_{z-1}} \sum_{\ell=1}^{z-1} |C_\ell| \cdot \alpha^{z-1-\ell} = \varepsilon,$$

where the first inequality follows by $H_v \subseteq H_{c_{z-1}}$. □

We are only left with showing that the estimates are well-behaved also along long paths. Before proving this, we show an auxiliary property that applies specifically to paths going from a cluster center to its descendants.

Lemma 26. *For $\varepsilon, \alpha, \delta \in (0,1)$, let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be the output of* `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$*, and suppose that each call to* `EstimateRatio` *as well as the call to* `ClusterSort` *is successful. Consider any connected component $\mathcal{C}$ in forest F and let i be the largest index such that $c_i \in \mathcal{C}$. If c_x is an ancestor of c_y in the tree $\mathcal{C}$ rooted at c_i, then $r(P(c_x, c_y)) \geq \frac{1}{\alpha^{x-y}}$.*

Proof. By construction, we must have $x \geq y$. Let $P(c_x, c_y) := c_x = c_{i_1}, \ldots, c_{i_k} = c_y$ be the unique path from c_x to c_y. Note that, by construction, for each $j \in [k-1]$, $r(c_{i_j}, c_{i_{j+1}}) \geq \frac{1}{\alpha^{i_j - i_{j+1}}}$. Thus,

$$r(P(c_x, c_y)) = \prod_{j=1}^{k-1} r(c_{i_j}, c_{i_{j+1}}) \geq \prod_{j=1}^{k-1} \frac{1}{\alpha^{i_j - i_{j+1}}} = \frac{1}{\alpha^{i_1 - i_k}} = \frac{1}{\alpha^{x-y}}.$$ □

We are now ready to show that the estimates are well-behaved on long paths. This concludes the proof of the second point of Definition 4 and hence all four properties of Definition 4 are proved.

Lemma 27. *For $\varepsilon, \alpha, \delta \in (0,1)$, let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be the output of* `BuildEstimationForest`$(\alpha, \varepsilon, \delta)$*, and suppose that each call to* `EstimateRatio` *as well as the call to* `ClusterSort` *is successful. Suppose that $u, v \in [n]$ are in the same connected component $\mathcal{C}$, and $\gamma(u) > \gamma(v)$ and $d(u,v) > 5$. Then, $\sum_{s \in K_v} r(P(s,u)) \le \varepsilon$, where $K_v = \{s \in \mathcal{C} \mid \gamma(s) \le \gamma(v)\}$.*

Proof. Let $R \in [T]$ be the maximum index such that $c_R \in \mathcal{C}$, and consider the tree $\mathcal{C}$ rooted at c_R (see Figure 4 for reference). Since $d(u,v) \ge 6$, it must be $d(c_{\gamma(u)}, c_{\gamma(v)}) \ge 4$. Similarly to Lemma 25, let c_z be the ancestor of $c_{\gamma(v)}$ at distance 2 from $c_{\gamma(v)}$ in F. Note that c_z exists and $\gamma(u) \ge z$ since $d(c_{\gamma(u)}, c_z) \ge 2$. Let c_x be the sibling of $c_{\gamma(u)}$ with minimum cluster index (possibly, $c_x = c_{\gamma(u)}$ or it might also be $c_x = c_z$; but it surely holds $x \le \gamma(u)$). Note that $d(u, c_x) \le 3$ and c_x is an ancestor of c_z (hence, $x \ge z$). Thus, by Lemma 24 and Lemma 26,

$$\begin{aligned} r(P(u,c_z)) &= r(P(u,c_x)) \cdot r(P(c_x,c_z)) \ge (1-\varepsilon_1)^3 \frac{w_u}{w_{c_x}} \cdot \frac{1}{\alpha^{x-z}} \\ &\ge (1-\varepsilon_1)^3 \frac{w_u}{w_{c_{\gamma(u)}}} \cdot \frac{w_{c_{\gamma(u)}}}{w_{c_x}} \ge (1-\varepsilon_1)^3 \cdot \frac{\alpha}{2} \cdot \frac{1}{\alpha^{\gamma(u)-x}} \ge (1-\varepsilon_1)^3 \cdot \frac{\alpha}{2}. \end{aligned} \tag{10}$$

Again similarly to Lemma 25, there must exists c_k, with $z > k \ge \gamma(v)$, such that $r(c_z, c_k) = \infty$. Note that $d(c_z, c_k) = 2$ and, in particular, c_k and $c_{\gamma(v)}$ are siblings (or $c_k = c_{\gamma(v)}$). Let $K_{c_k} = \mathcal{C} \cap \left(\bigcup_{\ell=1}^{k} C_\ell\right)$. Note that $K_v = \mathcal{C} \cap \left(\bigcup_{\ell=1}^{\gamma(v)} C_\ell\right) \subseteq K_{c_k}$. Consider any $s \in K_{c_k}$, by the same argument as in Lemma 25,

$$w_{c_z} \ge \frac{\alpha}{2 \cdot \beta_k \cdot \alpha^{k-\gamma(s)}} \cdot w_s. \tag{11}$$

Consider now $A = \{s \in K_{c_k} \mid c_{\gamma(s)} \text{ is a sibling of } c_k\}$. Note that any vertex in A is at distance either 2 or 3 from c_z. Thus, for any $a \in A$, by Lemma 24,

$$r(P(c_z,a)) \ge (1-\varepsilon_1)^3 \cdot \frac{w_{c_z}}{w_a} \overset{(11)}{\ge} \frac{(1-\varepsilon_1)^3 \alpha}{2 \cdot \beta_k \cdot \alpha^{k-\gamma(a)}}. \tag{12}$$

Let ψ be the smallest index such that $c_\psi \in A$. Let $B = K_{c_k} \setminus A$. Note that for any $b \in B$, c_ψ is an ancestor of b. Then for any $b \in B$, by Lemma 26,

$$r(P(c_\psi, b)) \ge \frac{1}{\alpha^{\psi - \gamma(b)}}. \tag{13}$$

Thus, for any $b \in B$,

$$r(P(c_z,b)) = r(P(c_z,c_\psi)) \cdot r(P(c_\psi,b)) \overset{(12),(13)}{\ge} \frac{(1-\varepsilon_1)^3\alpha}{2 \cdot \beta_k \cdot \alpha^{k-\psi}} \cdot \frac{1}{\alpha^{\psi-\gamma(b)}} = \frac{(1-\varepsilon_1)^3\alpha}{2 \cdot \beta_k \cdot \alpha^{k-\gamma(b)}}. \tag{14}$$

Therefore, for any $s \in K_{c_k}$,

$$\begin{aligned} r(P(u,s)) &= r(P(u,c_z)) \cdot r(P(c_z,s)) \overset{(10),(12),(14)}{\ge} \frac{(1-\varepsilon_1)^3\alpha}{2} \cdot \frac{(1-\varepsilon_1)^3\alpha}{2 \cdot \beta_k \cdot \alpha^{k-\gamma(s)}} \\ &= \frac{(1-\varepsilon_1)^6\alpha^2}{4 \cdot \beta_k \cdot \alpha^{k-\gamma(s)}} \ge \frac{\alpha^2}{8 \cdot \beta_k \cdot \alpha^{k-\gamma(s)}} \ge \frac{Z_k}{\varepsilon \cdot \alpha^{k-\gamma(s)}}, \end{aligned} \tag{15}$$

where we used that $\varepsilon_1 \le \frac{1}{10}$ and $(1-x)^6 \ge 1/2$ for all $x \le \frac{1}{10}$. Note that $r(P(s,u)) = \frac{1}{r(P(u,s))}$. Therefore,

$$\sum_{s \in K_v} r(P(s,u)) \overset{(15)}{\le} \sum_{s \in K_{c_k}} \frac{\varepsilon \cdot \alpha^{k-\gamma(s)}}{Z_k} \le \frac{\varepsilon}{Z_k} \sum_{\ell=1}^{k} |C_\ell| \cdot \alpha^{k-\ell} = \frac{\varepsilon}{Z_k} \cdot Z_k = \varepsilon,$$

where we used that $K_v \subseteq K_{c_k} \subseteq \bigcup_{\ell=1}^{k} C_\ell$. □

We now have all the ingredients to prove that Algorithm 4 produces a $(5, \varepsilon)$-estimation-forest.

Proof of Theorem 21. By Theorem 20, `ClusterSort`$(\alpha, \varepsilon_1, \delta/3)$ correctly returns a $(\frac{2}{\alpha}, \frac{1}{\alpha}, \varepsilon_1)$-cluster graph with probability at least $1 - \frac{\delta}{3}$. Moreover, the next part of the algorithm makes at most $2n$ calls to `EstimateRatio`, and each call is successful with probability at least $1 - \frac{\delta}{6n}$. Therefore, all the `EstimateRatio` calls are successful with probability at least $1 - \frac{\delta}{3}$ and therefore each call to `EstimateRatio` as well as the call to `ClusterSort` is successful with probability at least $1 - \frac{2}{3} \cdot \delta$. If this event happens, then Lemmas 23 to 25 and 27 ensure that the algorithm returns a $(5, \varepsilon)$-estimation-forest. Finally, with probability at least $1 - \frac{\delta}{3}$, the upper bound on the number of queries follows by Lemma 22. □

6.3 Learning the MNL from the Estimation-Forest

In this section, we show how to use a (t, ε)-estimation-forest to produce MNL weights that approximate the hidden MNL within a d_{mul} error of $O(\varepsilon)$.

Intuitively, an estimation-forest ensures that multiplying the estimates along a path gives a good estimate for the ratio of the weights of the two endpoints. These estimates are, in some sense, well-behaved even if the path is long. These properties suggest the following natural algorithm to generate the weights from a tree of the estimation-forest: assign an arbitrary weight to an initial vertex, and then assign all the other weights following the unique path from the initial one to all of the others. If we have multiple trees, by the properties of the estimation-forest it must be that any item in the tree with larger cluster indices wins against all the items of the other tree with very large probability (at least $1 - \varepsilon$). To mimic this property with our estimated weights, we rescale our estimates for the second tree by a sufficiently small value. The algorithm boils down to a depth-first search that we present in Algorithm 6.

Algorithm 5 `GenerateWeightsRec`$(F, r, v, parent, \hat{w})$

Input: A forest $F = ([n], E)$ with a directed weighting r, the current vertex $v \in [n]$, the parent $parent \in [n]$ of v, and a vector of values $\hat{w}_1, \ldots, \hat{w}_n$.
Output: For each i in the subtree of v, it sets $\hat{w}_i$ to a positive weight, and it returns a set W containing all the indices that have been modified.
$W = \varnothing$
for $\{u, v\} \in E$ such that $u \neq parent$ **do**
    $\hat{w}_u = \hat{w}_v \cdot r(u, v)$
    $W = W \cup \{u\} \cup$ `GenerateWeightsRec`$(F, r, u, v, \hat{w})$
return W

We now show that the ratios of the estimated weights are well-behaved.

Lemma 28. *Let $\varepsilon \in (0, \frac{1}{3})$, $\alpha \in (0, 1)$, and let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be a (t, ε)-estimation-forest for MNL M and let $\hat{M}$ be the MNL returned by* `GenerateWeights`$(\mathcal{F})$*. The following holds:*

1. *for any $u, v \in [n]$ such that $d(u, v) \leq t$, $\frac{\hat{w}_u}{\hat{w}_v} \in (1 \pm \varepsilon) \cdot \frac{w_u}{w_v}$,*
2. *for any $u, v \in [n]$ such that $d(u, v) > t$ and $\gamma(u) > \gamma(v)$, it holds that $\sum_{s \in H_v} \frac{\hat{w}_s}{\hat{w}_u} \leq 2\varepsilon$, where $H_v = \{s \in [n] \mid \gamma(s) \leq \gamma(v)\}$.*

Algorithm 6 `GenerateWeights`($\mathcal{F}$)

Input: A (t,ε)-estimation-forest $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ for an MNL M.
Output: A new MNL $\hat{M}$ such that $d_1(M, \hat{M}) \leq 9 \cdot \varepsilon$.
Initialize an array $\hat{w} = (\hat{w}_1, \ldots, \hat{w}_n)$, setting each entry to $\perp$
$w_{\min} = 1$
for $c = c_T, c_{T-1} \ldots, c_1$ **do** ▷ iterate over the centers from the one of largest weight
  if $\hat{w}_c = \perp$ **then**
    $\hat{w}_c = 1$ ▷ initially assign weights w.r.t. 1
    $W = \{c\} \cup$ `GenerateWeightsRec`$(F, r, c, -1, \hat{w})$
    $\Upsilon = \max_{j \in W} \{\hat{w}_j\}$
    if $c \neq c_T$ **then**
      for $j \in W$ **do** ▷ rescale weights to account for the trees with larger weights
        $\hat{w}_j = \hat{w}_j \cdot \frac{\varepsilon}{\Upsilon \cdot n} \cdot w_{\min}$
    $w_{\min} = \min \left\{ w_{\min}, \min_{j \in W} \{\hat{w}_j\} \right\}$
return MNL $\hat{M}$ induced by weights $\hat{w}_1, \ldots, \hat{w}_n$

Proof. Let us start with the first point. Since $d(u,v) \leq t$, u and v are in the same connected component $\mathcal{C}$. Let $i \in [T]$ be the maximum index such that $c_i \in \mathcal{C}$. By construction, for any $x \in \mathcal{C}$, $\hat{w}_x = r(P(x, c_i)) \cdot \hat{w}_{c_i}$. Thus, we have,

$$\frac{\hat{w}_u}{\hat{w}_v} = \frac{r(P(u,c_i)) \cdot \hat{w}_{c_i}}{r(P(v,c_i)) \cdot \hat{w}_{c_i}} = r(P(u,c_i)) \cdot r(P(c_i,v)) = r(P(u,v)), \tag{16}$$

where the last equality holds since $\mathcal{C}$ is a tree and there is a unique path between every pair of vertices. Therefore, since $\mathcal{F}$ is a (t,ε)-estimation-forest, $\frac{\hat{w}_u}{\hat{w}_v} = r(P(u,v)) \in (1 \pm \varepsilon) \cdot \frac{w_u}{w_v}$.

We now prove the second point. Fix any $u, v \in [n]$ such that $d(u,v) > t$ and $\gamma(u) > \gamma(v)$. Let $\mathcal{C}$ be the connected component of u. Let us partition $H_v = \{s \in [n] \mid \gamma(s) \leq \gamma(v)\}$ into $A = \{s \in \mathcal{C} \mid \gamma(s) \leq \gamma(v)\}$ and $B = H_v \setminus A$. Observe that either A is empty, or $v \in A$, by Definition 4. Consider any $s \in A$. Since s and u are in the same connected component, by Equation (16), $\frac{\hat{w}_s}{\hat{w}_u} = r(P(s,u))$. Then, by Definition 4,

$$\sum_{s \in A} \frac{\hat{w}_s}{\hat{w}_u} \overset{(16)}{=} \sum_{s \in A} r(P(s,u)) \leq \varepsilon.$$

Consider now $s \in B$ and let j be the largest index such that c_j is in the same connected component as s. By construction of $\hat{M}$, it must be the case that

$$\hat{w}_{c_j} = w_{\min} \cdot \frac{\varepsilon}{\Upsilon \cdot n}, \tag{17}$$

for some $w_{\min} \leq \hat{w}_u$ given that $\gamma(u) > j$ and u and c_j are in different trees. Moreover, by the definition of Υ, $r(P(s,c_j)) \leq \Upsilon$. Observe that $\hat{w}_s = r(P(s,c_j)) \cdot \hat{w}_{c_j}$, and thus,

$$\hat{w}_s = r(P(s,c_j)) \cdot \hat{w}_{c_j} \leq \Upsilon \cdot \hat{w}_{c_j} \overset{(17)}{\leq} \frac{\varepsilon}{n} \cdot \hat{w}_u.$$

Since this holds for each $s \in B$,

$$\sum_{s \in B} \frac{\hat{w}_s}{\hat{w}_u} \leq \sum_{s \in B} \frac{\varepsilon}{n} = \varepsilon \cdot \frac{|B|}{n} \leq \varepsilon.$$

Thus, $\sum_{s\in H_v} \frac{\hat{w}_s}{\hat{w}_u} \le 2\varepsilon$. □

We can finally prove that $\hat{M}$ has the desired guarantees.

Theorem 29. *Let $\varepsilon \in (0, \frac{1}{9})$, $t \ge 2$, and let $\mathcal{F} = (F = ([n], E), r, (C_1, \dots, C_T), (c_1, \dots, c_T))$ be a (t, ε)-estimation-forest for MNL M and let $\hat{M}$ be the MNL returned by* `GenerateWeights`$(\mathcal{F})$. *Then, $d_{mul}(M, \hat{M}) \le 6\varepsilon$.*

Proof. Consider any slate $S \subseteq [n]$. Let $m \in S$ be such that for any other $s \in S$, $\gamma(s) \le \gamma(m)$. Let $S_1 = \{s \in S \mid d(s, m) \le t\}$ and $S_2 = S \setminus S_1$. Let $m_2 \in S_2$ be such that for any $s \in S_2$, $\gamma(s) \le \gamma(m_2)$. Let $H_{m_2} = \{s \in [n] \mid \gamma(s) \le \gamma(m_2)\}$.

By definition of S_2, $d(m, m_2) > t$. Note that, by Definition 4, this implies $\gamma(m_2) \ne \gamma(m)$. By definition of m_2 and m, this in turn implies $\gamma(m_2) < \gamma(m)$.

Note also that $S_2 \subseteq H_{m_2}$ by the choice of m_2. Given these observations, by Definition 4, we have,

$$\sum_{s\in S_2} M_S(s) \le \sum_{s\in S_2} \frac{w_s}{w_m} \le \sum_{s\in H_{m_2}} \frac{w_s}{w_m} \le \varepsilon. \tag{18}$$

Similarly, by Lemma 28,

$$\sum_{s\in S_2} \hat{M}_S(s) \le \sum_{s\in S_2} \frac{\hat{w}_s}{\hat{w}_m} \le \sum_{s\in H_{m_2}} \frac{\hat{w}_s}{\hat{w}_m} \le 2\varepsilon. \tag{19}$$

Let us now focus on S_1. Note that, for any $u, v \in S_1$, $\frac{\hat{w}_u}{\hat{w}_v} \in (1\pm 3\varepsilon)\frac{w_u}{w_v}$. Specifically, by Lemma 28, $\frac{\hat{w}_u}{\hat{w}_m} \in (1 \pm \varepsilon)\frac{w_u}{w_m}$ and $\frac{\hat{w}_m}{\hat{w}_v} \in (1 \pm \varepsilon)\frac{w_m}{w_v}$, thus,

$$(1 - 2\varepsilon) \cdot \frac{w_u}{w_v} \le (1 - \varepsilon)^2 \cdot \frac{w_u}{w_v} \le \frac{\hat{w}_u}{\hat{w}_m} \cdot \frac{\hat{w}_m}{\hat{w}_v} \le (1 + \varepsilon)^2 \cdot \frac{w_u}{w_v} \le (1 + 3\varepsilon) \cdot \frac{w_u}{w_v}, \tag{20}$$

where we used that $(1 + \varepsilon)^2 \le 1 + 3\varepsilon$, for $\varepsilon \in (0, 1)$, and $(1 - \varepsilon)^2 \ge 1 - 2\varepsilon$ for $\varepsilon > 0$.

Note that by Equation (18) and Equation (19), we also know that:

$$\sum_{s\in S_2} w_s \le \varepsilon \cdot w_m, \quad \text{and} \quad \sum_{s\in S_2} \hat{w}_s \le 2\varepsilon \cdot \hat{w}_m. \tag{21}$$

Therefore, for any $v \in S_1$, we have,

$$\begin{aligned}
\hat{M}_S(v) = \frac{1}{\sum_{s\in S} \frac{\hat{w}_s}{\hat{w}_v}} &\le \frac{1}{\sum_{s\in S_1} \frac{\hat{w}_s}{\hat{w}_v}} \overset{(20)}{\le} \frac{1}{(1-2\varepsilon)\sum_{s\in S_1} \frac{w_s}{w_v}} = \frac{1}{(1-2\varepsilon)\left(\sum_{s\in S} \frac{w_s}{w_v} - \sum_{s\in S_2} \frac{w_s}{w_v}\right)} \\
&\overset{(21)}{\le} \frac{1}{(1-2\varepsilon)\left(\sum_{s\in S} \frac{w_s}{w_v} - \varepsilon\frac{w_m}{w_v}\right)} \le \frac{1}{(1-2\varepsilon)\left(\sum_{s\in S} \frac{w_s}{w_v} - \varepsilon\sum_{s\in S} \frac{w_s}{w_v}\right)} \\
&= \frac{1}{(1-2\varepsilon)(1-\varepsilon)\sum_{s\in S} \frac{w_s}{w_v}} \le \frac{1}{(1-3\varepsilon)\sum_{s\in S} \frac{w_s}{w_v}} \\
&\le (1+6\varepsilon)\frac{1}{\sum_{s\in S} \frac{w_s}{w_v}} = (1+6\varepsilon) \cdot M_S(v),
\end{aligned}$$

where we used that $(1-2\varepsilon)(1-\varepsilon) \geq 1-3\varepsilon$ for $\varepsilon > 0$ and $\frac{1}{1-a} \leq 1+2a$ for $a \in (0, \frac{1}{2})$. Similarly,

$$\begin{aligned}
\hat{M}_S(v) &= \frac{1}{\sum_{s\in S} \frac{\hat{w}_s}{\hat{w}_v}} = \frac{1}{\sum_{s\in S_1} \frac{\hat{w}_s}{\hat{w}_v} + \sum_{s\in S_2} \frac{\hat{w}_s}{\hat{w}_v}} \overset{(21)}{\geq} \frac{1}{\sum_{s\in S_1} \frac{\hat{w}_s}{\hat{w}_v} + 2\varepsilon \cdot \frac{\hat{w}_m}{\hat{w}_v}} \\
&\geq \frac{1}{\sum_{s\in S_1} \frac{\hat{w}_s}{\hat{w}_v} + 2\varepsilon \cdot \sum_{s\in S_1} \frac{\hat{w}_s}{\hat{w}_v}} = \frac{1}{(1+2\varepsilon)\sum_{s\in S_1} \frac{\hat{w}_s}{\hat{w}_v}} \overset{(20)}{\geq} \frac{1}{(1+2\varepsilon)(1+3\varepsilon)\sum_{s\in S_1} \frac{w_s}{w_v}} \\
&\geq \frac{1}{(1+6\varepsilon)\sum_{s\in S_1} \frac{w_s}{w_v}} \geq (1-6\varepsilon)\cdot M_{S_1}(v) \geq (1-6\varepsilon)\cdot M_S(v),
\end{aligned}$$

where we used that $(1+2\varepsilon)(1+3\varepsilon) \leq 1+6\varepsilon$ for $\varepsilon \in (0, \frac{1}{6})$ and $\frac{1}{1+a} \geq 1-a$ for $a \geq 0$. Thus, $\hat{M}_S(v) \in (1 \pm 6\varepsilon) M_S(v)$ for any $v \in S_1$.

Since this holds for any slate, we have that $d_{\text{mul}}(M, \hat{M}) \leq 6\varepsilon$. □

Putting everything together we obtain our main result as a corollary. The proof simply consists in first building an estimation-forest and then extracting the weights from it. We also show how to deal with the fact that weights could be large to store. Note that Theorem 6 is a special case of the following result when $\delta = n^{-c}$ for some constant c.

Theorem 30. *There exists an adaptive randomized algorithm that, takes as input $\varepsilon \in (0,1)$, $\delta \in (0,1)$, and access to* Sample *oracle for an MNL M supported on $[n]$ for $n \in \mathbb{N}$, and with probability at least $1-\delta$, makes $O\left(n \log(\frac{n}{\delta}) \cdot \left(\frac{1}{\varepsilon^3} + \frac{\log(1/\delta)}{\log n}\right)\right)$* Sample *queries and solves the MNL learning problem for d_{mul}-error with accuracy parameter ε. The algorithm runs in time proportional to the query complexity.*

Proof. Let $\varepsilon' = \frac{\varepsilon}{20}$. Obtain a $(5, \frac{\varepsilon'}{6})$-estimation-forest $\mathcal{F}$ by calling `BuildEstimationForest`$(\frac{1}{2}, \frac{\varepsilon'}{6}, \delta)$. Note that the algorithm has the desired query complexity. Then, obtain the MNL $\hat{M}$ by running algorithm `GenerateWeights`$(\mathcal{F})$. By Theorem 29, we have $d_{\text{mul}}(M, \hat{M}) \leq \varepsilon'$.

We now focus on the computational complexity. Note that Algorithm 4 has a running time equal to its query complexity. Also, Algorithm 6 performs $O(n)$ multiplications and makes no further queries. Let us assume without loss of generality that the output weights $\hat{w}_1, \ldots, \hat{w}_n$ are sorted so that $\hat{w}_1 \geq \cdots \geq \hat{w}_n$. Note that:

$$\hat{w}_i \leq \hat{w}_{i+1} \cdot \frac{300 \cdot n}{\varepsilon'}. \tag{22}$$

This is because each estimated weight is separated by its adjacent weight in the order by either: (i) a factor of $\frac{n}{\varepsilon'}$ as in Line 12 of Algorithm 6, or (ii) a factor of $r(u,v)$ as in Line 5 of Algorithm 5. In the second case, by the properties of `EstimateRatio` (Lemma 16) and by inspecting the pseudocode of Algorithm 4, each value $r(u,v)$ computed by `BuildEstimationForest`$(\frac{1}{2}, \frac{\varepsilon'}{9}, \delta)$ is at most $(1 + \frac{\varepsilon'}{10}) \cdot \frac{288 \cdot n}{\varepsilon'} \leq \frac{300 \cdot n}{\varepsilon'}$. Note that storing all the values $\{\hat{w}_i\}_{i\in[n]}$ directly would require $O(n^2 \log \frac{n}{\varepsilon})$ bits.

In order to address this issue, instead of storing the values $\{\hat{w}_i\}_{i\in[n]}$, we store their natural logarithm approximately and compute directly on these values. In particular, for any number x used in the algorithm, we maintain a value $\lambda(x)$ that approximates $\ln(x)$. For all the values $x = r(u,v)$, which was represented as a fraction and computed using standard (exact) arithmetic by the previous subroutines, we compute a value $\lambda(x)$ such that:

$$\lambda(x) \in \ln x \pm \frac{\varepsilon'}{n^2}.$$

We do the same for the value $x = \frac{\varepsilon'}{n}$. In general, for any number x, this value can be represented with $O(\ln \ln x + \ln \frac{n}{\varepsilon})$ bits. Whenever the algorithm needs to multiply two numbers x and y to obtain some $z = x \cdot y$ (e.g., Line 5 of `GenerateWeightsRec` or Line 12 of `GenerateWeights`), we instead compute $\lambda(z) = \lambda(x) + \lambda(y)$. Similarly, for $z = \max\{x, y\}$, we compute $\lambda(z) = \max\{\lambda(x), \lambda(y)\}$.

Note that every value $\frac{1}{\Upsilon}$ is determined by the product of at most n numbers, and therefore $\lambda(\frac{1}{\Upsilon})$ is correct within an additive error of $\frac{\varepsilon'}{n}$. Now, each weight is computed as the product of at most $2n$ numbers (considering the values of $r(u, v)$, $\frac{1}{\Upsilon}$, and $\frac{\varepsilon'}{n}$), and for each of these numbers x in the product, the value $\lambda(x)$ is correct within an additive error of at most $\frac{\varepsilon'}{n}$. Therefore, $\lambda(\hat{w}_i)$ is correct within a $2\varepsilon'$ additive error. This implies that:

$$(1 - 4\varepsilon') \cdot \hat{w}_u \leq e^{-2\varepsilon'} \cdot \hat{w}_u \leq e^{\ln(\hat{w}_u) - 2\varepsilon'} \leq e^{\lambda(\hat{w}_u)} \leq e^{\ln(\hat{w}_u) + 2\varepsilon'} \leq e^{2\varepsilon'} \cdot \hat{w}_u \leq (1 + 4\varepsilon') \cdot \hat{w}_u \tag{23}$$

The algorithm then outputs the approximate logarithms of the weights $\{\lambda(\hat{w}_i)\}_{i \in [n]}$. If one were to use the values $\{e^{\lambda(\hat{w}_i)}\}_{i \in [n]}$ as proxies for the weights $\{\hat{w}_i\}_{i \in [n]}$ these would be correct to within multiplicative error $4\varepsilon'$ (by (23)). In particular, we have $d_{\mathrm{mul}}(M, \tilde{M}) \leq 18\varepsilon'$, where $\tilde{M}$ is the MNL supported on $\{e^{\lambda(\hat{w}_i)}\}_{i \in [n]}$.

With these changes, all arithmetic operations performed need to be executed on numbers of at most $O(\log \frac{n}{\varepsilon})$ bits (by (22)), and thus each of them can be executed in time $O(\log \frac{n}{\varepsilon})$. Therefore, the runtime of $O(n \log \frac{n}{\varepsilon})$ of `GenerateWeights`$(\mathcal{F})$ is no larger than the query complexity. □

Supporting items of weight zero. In the context of MNLs, weights are assumed to be strictly positive. However, in the conditional sampling literature it is common to allow items of weight zero, and if a slate consists only of items of weight zero its distribution is uniform (Canonne, 2020). Our algorithm can be run as is and it works even in this case. Specifically, it turns out that any algorithm that can learn MNLs can also learn MNLs where items of zero weight are allowed. This is because these latter models arise as limits of MNLs and any algorithm that learns MNLs must necessarily learn these limiting models too, as we prove in Section E.

7 Adaptive $O(\frac{n \log n}{\varepsilon^2})$ Algorithm for d_1

In this section we discuss how to improve the dependence in ε when approximating the d_1-error instead of d_{mul}. The high level idea is still to build an estimation forest where edges are labeled with the ratio of the endpoints, but we relax the guarantees required by this structure. Specifically, our goal is to construct an ε-weak-estimation-forest as defined in Definition 7.

7.1 Constructing an ε-Weak-Estimation-Forest

We now show how to build an ε-weak-estimation-forest in $O(n \log n/\varepsilon^2)$ time. We will start with the clustering phase of Algorithm 3, and then we will build a path over the cluster centers (from the heaviest one to the lightest one), breaking the path when the edge does not require accurate estimate. We will use a potential similar to the one used in Algorithm 4 to determine when to create a new connected component.

We will make use of the following subroutine to estimate weight ratios so that one side of the inequality is without error. The guarantees of this subroutine follow by Bernstein's inequality.

Lemma 31 (`EstimateOneSidedRatio` Guarantees)**.** *Let $i, j \in [n]$ satisfy $w_i \geq w_j$. Let $Z \geq 1$, $\varepsilon, \delta \in (0, 1)$. With probability at least $1 - \delta$,* `EstimateOneSidedRatio`$(i, j, Z, \varepsilon, \delta)$ *returns a value $r(i, j)$ satisfying the following guarantees:*

Algorithm 7 `EstimateOneSidedRatio`$(i, j, Z, \varepsilon, \delta)$

Input: Two items $i, j \in [n]$ with $w_i \geq w_j$, parameters $Z \geq 1$, $\varepsilon, \delta \in (0, \frac{1}{2})$, and access to a Sample oracle for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$.
Output: An estimate $r(i,j) \in (0, \infty]$ of w_i/w_j.
$L = \ln \frac{2}{\delta}$
$m = \left\lceil \frac{512 \cdot Z \cdot L}{\varepsilon^2} \right\rceil$
$X = 0$
for $t = 1, \ldots, m$ **do**
    if Sample$(\{i,j\}) = j$ **then**
        $X = X + 1$
if $X < 64 \cdot L$ **then**
    return ∞
$\hat{p} = \frac{X - 4\sqrt{X \cdot L}}{m}$
return $\frac{1-\hat{p}}{\hat{p}}$

1. *If $r(i,j) \neq \infty$, then:*
$$\left(1 - \varepsilon \cdot \sqrt{\frac{w_i}{Z \cdot w_j}}\right) \frac{w_j}{w_i} \leq \frac{1}{r(i,j)} \leq \frac{w_j}{w_i}.$$
2. *If $r(i,j) = \infty$, then:*
$$\frac{w_j}{w_i} \leq \frac{\varepsilon^2}{2 \cdot Z}.$$

Moreover, the algorithm makes $O\left(\frac{Z \cdot \log(1/\delta)}{\varepsilon^2}\right)$ pairwise queries.

We will make use of the following elementary inequality:

Lemma 32. *For any $x_1, \ldots, x_n \in [0,1]$ and $z \in \mathbb{R}$, we have:*
$$\left|1 - z \cdot \prod_{i=1}^{n} x_i\right| \leq |1 - z| + \sum_{i=1}^{n} (1 - x_i).$$

We now analyze Algorithm 8 and show that it successfully builds a weak-estimation-forest.

Theorem 33 (Building a weak-estimation-forest)**.** *Let $\varepsilon, \alpha, \delta \in (0,1)$ with constant α, then, with probability at least $1 - \delta$,* `BuildWeakEstimationForest`$(\alpha, \varepsilon, \delta)$ *makes $O\left(n \log(\frac{n}{\delta}) \cdot \left(\frac{1}{\varepsilon^2} + \frac{\log(1/\delta)}{\log n}\right)\right)$* Sample *queries and returns an ε-weak-estimation-forest.*

Proof. Note that since we make $O(n)$ calls to `EstimateOneSidedRatio`, all calls to this subroutine as well as the call to `ClusterSort` are successful with probability larger than $1 - \delta$. Note that if `ClusterSort` is successful, then $w_{c_{j+1}} \geq \frac{1}{\alpha} w_{c_j} > w_{c_j}$, therefore the calls to `EstimateOneSidedRatio` are well defined. We condition on these events for the rest of the proof. Note also that the query complexity follows by the complexity of `ClusterSort` (Proposition 19) as well as by Lemma 31:
$$\sum_{j=1}^{T-1} O\left(\frac{Z_j}{\eta^2} \cdot \log\left(\frac{n}{\delta}\right)\right) = \frac{\log(n/\delta)}{\varepsilon^2} \cdot O\left(\sum_{j=1}^{T-1} Z_j\right) \overset{(4)}{=} O\left(\frac{n \log(n/\delta)}{\varepsilon^2}\right).$$

We now focus on proving that the properties of Definition 7 are satisfied.

Algorithm 8 `BuildWeakEstimationForest`$(\alpha, \varepsilon, \delta)$

Input: Parameters $\alpha, \varepsilon, \delta \in (0,1)$, and access to a Sample oracle for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$.
Output: An ε-weak-estimation-forest $\mathcal{F} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$.
$K_\alpha = 16\left(1 + \frac{2}{\alpha(1-\sqrt{\alpha})}\right)$
$\eta = \frac{\varepsilon}{K_\alpha}$
$(F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T)) = \texttt{ClusterSort}(\alpha, \eta, \frac{\delta}{3})$
$Z_0 = 0$
for $j = 1, \ldots, T$ **do**
    $Z_j = \alpha Z_{j-1} + |C_j|$
for $j = 1, \ldots, T-1$ **do**
    $r(c_{j+1}, c_j) = \texttt{EstimateOneSidedRatio}(c_{j+1}, c_j, Z_j, \eta, \frac{\delta}{3n})$
    if $r(c_{j+1}, c_j) \neq \infty$ **then**
        $E = E \cup \{\{c_j, c_{j+1}\}\}$
return $(F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$

We start by proving the second property. Let $u \in [n]$ and consider any other vertex $v \in [n]$ in a different component than u and such that $\gamma(u) \geq \gamma(v)$. Note that by construction, the connected component of the resulting graph consists of adjacent clusters, therefore for any u' in the connected component of u and v' in the connected component of v, we have $\gamma(u') > \gamma(v')$. Let j be the smallest index such that c_j is in the same connected component of u. By Lemma 31, since `EstimateOneSidedRatio` returned ∞, we have:

$$w_{c_j} \geq \frac{2 \cdot Z_{j-1}}{\eta^2} \cdot w_{c_{j-1}}.$$

Note that $\gamma(v) \leq j-1$. Consider any $s \in [n]$ such that $\gamma(s) \leq \gamma(v)$. We have:

$$w_u \overset{Definition\ 5}{\geq} \frac{\alpha}{2} \cdot w_{c_j} \geq \frac{\alpha \cdot Z_{j-1}}{\eta^2} \cdot w_{c_{j-1}} \overset{Definition\ 5}{\geq} \frac{\alpha^2 \cdot Z_{j-1} \cdot \alpha^{\gamma(s)-j+1}}{2 \cdot \eta^2} \cdot w_s \geq \frac{Z_{j-1} \cdot \alpha^{\gamma(s)-j+1}}{\varepsilon^2} \cdot w_s.$$

Therefore:

$$\sum_{\substack{s \in [n] \\ \gamma(s) \leq \gamma(v)}} \frac{w_s}{w_u} \leq \sum_{\substack{s \in [n] \\ \gamma(s) \leq \gamma(v)}} \frac{\varepsilon^2 \cdot \alpha^{j-1-\gamma(s)}}{Z_{j-1}} \leq \varepsilon^2 \cdot \sum_{\ell=1}^{j-1} \frac{|C_\ell| \cdot \alpha^{j-1-\ell}}{Z_{j-1}} = \varepsilon^2 \leq \varepsilon,$$

which concludes the proof of the second property of Definition 7.

We now focus on the first property. Fix any $u \in [n]$ and let $\mathcal{C}$ be its connected component. Consider any edge between two cluster centers in $\mathcal{C}$, say c_j and c_{j+1}. Since $r(c_{j+1}, c_j) \neq \infty$, by Lemma 31 we have:

$$1 - \eta \cdot \sqrt{\frac{w_{c_{j+1}}}{Z_j \cdot w_{c_j}}} \leq \frac{r(c_j, c_{j+1})}{w_{c_j}/w_{c_{j+1}}} \leq 1 \tag{24}$$

Consider any $s \in \mathcal{C}$ with $\gamma(s) \leq \gamma(u)$. We have:

$$r(P(s,u)) = r(s, c_{\gamma(s)}) \cdot r(c_{\gamma(u)}, u) \cdot \prod_{j=\gamma(s)}^{\gamma(u)-1} r(c_j, c_{j+1})$$

$$\iff \frac{r(P(s,u))}{w_s/w_u} = \frac{r(s,c_{\gamma(s)})}{w_s/w_{c_{\gamma(s)}}} \cdot \frac{r(c_{\gamma(u)},u)}{w_{c_{\gamma(u)}}/w_u} \cdot \prod_{j=\gamma(s)}^{\gamma(u)-1} \frac{r(c_j,c_{j+1})}{w_{c_j}/w_{c_{j+1}}}.$$

Define for convenience $\zeta_{s,u} := \frac{r(s,c_{\gamma(s)})}{w_s/w_{c_{\gamma(s)}}} \cdot \frac{r(c_{\gamma(u)},u)}{w_{c_{\gamma(u)}}/w_u}$. Note that $\zeta_{s,u} \in [(1-\eta)^2,(1+\eta)^2]$ since the edges $\{s,c_{\gamma(s)}\}$ and $\{u,c_{\gamma(u)}\}$ belong to the cluster graph. We then have:

$$\begin{aligned}
\left|\frac{w_s}{w_u} - r(P(s,u))\right| &= \frac{w_s}{w_u} \cdot \left|1 - \frac{r(P(s,u))}{w_s/w_u}\right| = \frac{w_s}{w_u} \cdot \left|1 - \zeta_{s,u} \cdot \prod_{j=\gamma(s)}^{\gamma(u)-1} \frac{r(c_j,c_{j+1})}{w_{c_j}/w_{c_{j+1}}}\right| \\
&\overset{\text{Lemma 32}}{\leq} \frac{w_s}{w_u}\left(|1-\zeta_{s,u}| + \sum_{j=\gamma(s)}^{\gamma(u)-1}\left(1 - \frac{r(c_j,c_{j+1})}{w_{c_j}/w_{c_{j+1}}}\right)\right) \\
&\overset{(24)}{\leq} \frac{w_s}{w_u}\left(3\eta + \eta \cdot \sum_{j=\gamma(s)}^{\gamma(u)-1} \sqrt{\frac{w_{c_{j+1}}}{Z_j \cdot w_{c_j}}}\right),
\end{aligned}$$

where the invocation of Lemma 32 is justified by (24).

Consider now any $A \subseteq \{s \in \mathcal{C} \mid \gamma(s) \leq \gamma(u)\}$. Without loss of generality, we assume $u \notin A$, since its contribution to the sum is zero. By the previous derivation, we have:

$$\begin{aligned}
\sum_{s\in A}\left|\frac{w_s}{w_u} - r(P(s,u))\right| &\leq \frac{3\eta}{w_u} \cdot \sum_{s\in A} w_s + \frac{\eta}{w_u} \sum_{s\in A}\sum_{j=\gamma(s)}^{\gamma(u)-1} w_s \cdot \sqrt{\frac{w_{c_{j+1}}}{Z_j \cdot w_{c_j}}} \\
&\leq \frac{3\eta}{w_u} \cdot \sum_{s\in A} w_s + \frac{\eta}{w_u} \sum_{j=1}^{\gamma(u)-1}\left(\sqrt{\frac{w_{c_{j+1}}}{Z_j \cdot w_{c_j}}} \cdot \sum_{\substack{s\in A \\ \gamma(s)\leq j}} w_s\right). \qquad (25)
\end{aligned}$$

Note that, by the properties of a cluster graph, for every $j < \gamma(u)$ we have:

$$\sum_{\substack{s\in A \\ \gamma(s)\leq j}} w_s \leq \sum_{\ell=1}^{j} |C_\ell| \cdot \frac{2}{\alpha} \cdot w_{c_\ell} \leq \frac{2 \cdot w_{c_j}}{\alpha} \cdot \sum_{\ell=1}^{j} |C_\ell| \alpha^{j-\ell} = \frac{2}{\alpha} \cdot Z_j \cdot w_{c_j},$$

from which we get:

$$\sum_{\substack{s\in A \\ \gamma(s)\leq j}} w_s = \sqrt{\sum_{\substack{s\in A \\ \gamma(s)\leq j}} w_s \cdot \sum_{\substack{s\in A \\ \gamma(s)\leq j}} w_s} \leq \sqrt{\frac{2}{\alpha} \cdot Z_j \cdot w_{c_j} \cdot \sum_{s\in A} w_s}.$$

Plugging this into (25), we obtain:

$$\begin{aligned}
\sum_{s\in A}\left|\frac{w_s}{w_u} - r(P(s,u))\right| &\leq \frac{3\eta}{w_u} \cdot \sum_{s\in A} w_s + \frac{\eta}{w_u} \sum_{j=1}^{\gamma(u)-1}\left(\sqrt{\frac{2 \cdot w_{c_{j+1}}}{\alpha} \cdot \sum_{s\in A} w_s}\right) \\
&\overset{\text{Definition 5}}{\leq} \frac{3\eta}{w_u} \cdot \sum_{s\in A} w_s + \frac{\eta}{w_u} \sum_{j=1}^{\gamma(u)-1}\left(\sqrt{\frac{4 \cdot w_u}{\alpha^2} \cdot \alpha^{\gamma(u)-j-1} \cdot \sum_{s\in A} w_s}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{3\eta}{w_u} \cdot \sum_{s \in A} w_s + \frac{2\eta}{\alpha} \cdot \left(\sqrt{\sum_{s \in A} \frac{w_s}{w_u}} \cdot \sum_{j=1}^{\gamma(u)-1} \alpha^{(\gamma(u)-j-1)/2} \right) \\
&\leq \frac{3\eta}{w_u} \cdot \sum_{s \in A} w_s + \frac{2\eta}{\alpha} \left(\frac{1 + \sum_{s \in A} w_s/w_u}{2} \cdot \sum_{j=0}^{\infty} (\sqrt{\alpha})^j \right) \\
&\leq \frac{3\eta}{w_u} \cdot \sum_{s \in A} w_s + \frac{\eta}{\alpha(1-\sqrt{\alpha})} \cdot \left(1 + \sum_{s \in A} \frac{w_s}{w_u} \right) \\
&\leq \eta \cdot \left(1 + \sum_{s \in A} \frac{w_s}{w_u} \right) \cdot \left(3 + \frac{1}{\alpha(1-\sqrt{\alpha})} \right) \\
&\leq \varepsilon \cdot \left(1 + \sum_{s \in A} \frac{w_s}{w_u} \right)
\end{aligned}
$$

where the third inequality uses that $\sqrt{x} \leq \frac{x+1}{2}$ for $x \geq 0$ and the last inequality follows by the definition of η.

This concludes the proof of the first property of Definition 7, and therefore the theorem. □

7.2 Learning the MNL from a Weak-Estimation-Forest

We now show that an ε-weak-estimation-forest is enough to obtain a d_1-error of $O(\varepsilon)$ which concludes the description of our algorithm. The algorithm to generate weights from the forest is exactly the same as Algorithm 6, only the analysis changes.

Theorem 34 (Reconstruction from a weak-estimation-forest)**.** *Let $\varepsilon \in (0, \frac{1}{9})$, and let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be an ε-weak-estimation-forest for MNL M and let $\hat{M}$ be the MNL returned by* `GenerateWeights`$(\mathcal{F})$*. Then, $d_1(M, \hat{M}) \leq 6\varepsilon$.*

Proof. Consider any slate S. Let $u \in S$ be an item such that for any other $s \in S$, $\gamma(u) \geq \gamma(s)$. Let $U \subseteq S$ be the items in S that are in the same connected component as u, and define $L = S \setminus U$.

Note that for any $i \in U$ we have $M_S(i) = (1 - \sum_{j \in L} M_S(j)) \cdot M_U(i)$ (and similarly for $\hat{M}_S(i)$). By this observation and by triangle inequality, we have:

$$
\|M_S - \hat{M}_S\|_1 \leq \|M_U - \hat{M}_U\|_1 + 2 \cdot \sum_{j \in L} (M_S(j) + \hat{M}_S(j)) \tag{26}
$$

We analyze the two terms in the sum separately. First, observe that if $L = \varnothing$, the second term is zero. Suppose instead that $L \neq \varnothing$ and let $\ell \in L$ be an item such that for any other $\ell_2 \in L$ it holds that $\gamma(\ell) \geq \gamma(\ell_2)$. By the second property of Definition 7 we have:

$$
\sum_{j \in L} M_S(j) \leq \sum_{\substack{s \in S \\ \gamma(s) \leq \gamma(\ell)}} M_S(s) \leq \sum_{\substack{s \in [n] \\ \gamma(s) \leq \gamma(\ell)}} \frac{w_s}{w_u} \overset{Definition\ 7}{\leq} \varepsilon. \tag{27}
$$

By construction of $\hat{w}_1, \ldots, \hat{w}_n$ in Algorithm 6, we have: $\hat{w}_s/\hat{w}_u \leq \frac{\varepsilon}{n}$ for any $s \in L$, given that $\gamma(u) \geq \gamma(s)$ and they are in different connected components. This implies:

$$
\sum_{j \in L} \hat{M}_S(j) \leq \sum_{j \in L} \frac{\hat{w}_j}{\hat{w}_u} \leq \varepsilon. \tag{28}
$$

Define a third MNL, M', induced by the weights:

$$w'_i := \frac{w_u}{\hat{w}_u} \cdot \hat{w}_i = w_u \cdot r(P(i,u)),$$

for $i \in U$. Since $w'_i \propto \hat{w}_i$, we have $M'_U(j) = \hat{M}_U(j)$ for each $j \in U$. By the first property of Definition 7 applied to $A = U \setminus \{u\}$, we have:

$$\begin{aligned}
&\sum_{s \in U \setminus \{u\}} \left| \frac{w_s}{w_u} - r(P(s,u)) \right| \leq \varepsilon \cdot \left(1 + \sum_{s \in U \setminus \{u\}} \frac{w_s}{w_u} \right) \\
\iff &\sum_{s \in U \setminus \{u\}} |w_s - w'_s| \leq \varepsilon \cdot \sum_{s \in U} w_s \\
\iff &\frac{\sum_{s \in U} |w_s - w'_s|}{\sum_{s \in U} w_s} \leq \varepsilon
\end{aligned}$$

We now use the following inequality. For any $a, b \in \mathbb{R}^n_{>0}$ we have:

$$\begin{aligned}
\sum_{i \in [n]} \left| \frac{a_i}{\sum_{j \in [n]} a_j} - \frac{b_i}{\sum_{j \in [n]} b_j} \right| &\leq \sum_{i \in [n]} \left| \frac{a_i}{\sum_{j \in [n]} a_j} - \frac{b_i}{\sum_{j \in [n]} a_j} \right| + \sum_{i \in [n]} \left| \frac{b_i}{\sum_{j \in [n]} a_j} - \frac{b_i}{\sum_{j \in [n]} b_j} \right| \\
&= \frac{\sum_{i \in [n]} |a_i - b_i|}{\sum_{j \in [n]} a_j} + \frac{\left| \sum_{j \in [n]} (b_j - a_j) \right|}{(\sum_{j \in [n]} a_j) \cdot (\sum_{j \in [n]} b_j)} \cdot \sum_{i \in [n]} b_i \\
&\leq 2 \cdot \frac{\sum_{i \in [n]} |a_i - b_i|}{\sum_{j \in [n]} a_j}.
\end{aligned}$$

Therefore, we have:

$$\|M_U - \hat{M}_U\|_1 = \|M_U - M'_U\|_1 = \sum_{i \in U} \left| \frac{w_i}{\sum_{j \in U} w_j} - \frac{w'_i}{\sum_{j \in U} w'_j} \right| \leq 2 \cdot \frac{\sum_{i \in U} |w_i - w'_i|}{\sum_{j \in U} w_j} \leq 2\varepsilon. \tag{29}$$

Putting together (26), (27), (28), and (29), we obtain: $\|M_S - \hat{M}_S\|_1 \leq 6\varepsilon$ for any slate S, concluding the proof. □

It is easy to see that Theorem 8 follows immediately from Theorem 34 and Theorem 33.

8 The Non-Adaptive Algorithm

In this section we show that we can learn an MNL within a d_{mul}-error of ε by making at most $O\left(\frac{n^2 \log(n/\varepsilon) \log(n/\delta)}{\varepsilon^3}\right)$ non-adaptive queries. Specifically, by Lemma 9, it is sufficient to prove Theorem 10, as this will imply the wanted result (Corollary 11). We recall the statement of Theorem 10.

Theorem 10. *Choose any $\varepsilon, \delta \in (0,1)$. There exists an adaptive randomized algorithm that, with probability at least $1 - \delta$, queries each pair at most $O\left(\frac{\log(n/\varepsilon) \cdot \log(n/\delta)}{\varepsilon^3}\right)$ times and solves the MNL Learning Problem for d_{mul}-error on $[n]$ with accuracy parameter ε.*

Recall that given a (t, ε)-estimation-forest, one can find the weights of an estimate MNL $\hat{M}$ with $d_{\text{mul}}(M, \hat{M}) \leq O(\varepsilon)$ without making any more Sample queries by using `GenerateWeights` (Algorithm 6). Therefore, our goal is to design an adaptive algorithm that queries each pair of items at most $O(\log^2 n)$ times (with potentially some dependency on the error parameters δ and ε) and then produces a (t, ε)-estimation-forest. Observe that, unfortunately, Algorithm 4 does not have this property for two reasons.

First, the algorithm used to compute the ε_o-ordering (described in Theorem 18) can compare some items $\Omega(\log^3 n)$ times. This can easily be fixed by creating the cluster graph via a variant of the classic Quicksort algorithm where each comparison is repeated sufficiently many times. In doing so, we obtain a smaller upper bound on the number of queries on each pair, in exchange for a higher overall worst-case query complexity. In fact, this algorithm will make $O(n \log^2 n)$ queries in total (for constant δ and ε), but each pair will be queried at most $O(\log n)$ times. Formally, we have the following result, that we prove in Section D.

Proposition 35. *There exists an algorithm* `QuicksortClustering`$(\alpha, \varepsilon, \delta)$ *that, given parameters* $\alpha, \varepsilon, \delta \in (0, 1)$ *and access to a* Sample *oracle for an MNL* M *supported on* $[n]$*, queries each pair of items at most* $O\left(\frac{\log(n/\delta)}{\alpha\varepsilon^2}\right)$ *times and that, with probability at least* $1 - \delta$*, returns a* $(\frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon)$*-cluster graph.*

The second reason is that Algorithm 4 might make $\Omega(n)$ queries to some pairs after the construction of the cluster graph. Indeed, consider an instance with $T = 3$ clusters with $|C_1| = n - 2$ and $|C_2| = |C_3| = 1$. For constant α and ε, we have $Z_2 = \Theta(n)$. In this case, Algorithm 4 queries the pair $\{c_3, c_2\}$ for $\Theta(Z_2) = \Theta(n)$ times and therefore does not have the property we seek to obtain an efficient non-adaptive algorithm.

We fix this issue by introducing two technical ingredients. First, we modify our algorithm that constructs the estimation-forest so that it requires at most $O(|C_j| \cdot \log^2(n))$ queries between any pair of cluster centers $\{c_i, c_j\}$. Second, instead of estimating the ratio w_{c_i}/w_{c_j} using only queries to the slate $\{c_i, c_j\}$, we make use of a new subroutine that constructs an estimate of w_{c_i}/w_{c_j} by querying each pair $\{c_i, e\}$, with $e \in C_j$, a balanced number of times. By dividing the $O(|C_j| \cdot \log^2(n))$ queries equally among the $|C_j|$ items of cluster C_j, we obtain an algorithm that queries each pair at most $O(\log^2(n))$ times.

We first show how to spread the queries over the cluster in Section 8.1, and then we show an algorithm to build the estimation-forest by querying each pair at most $O(\log^2 n)$ times in Section 8.2.

8.1 Spreading the Queries Among the Cluster Items

Algorithm 9 `GetGeometric`(u, v)

Input: Two items u and v in $[n]$ and access to a Sample oracle for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$.
Output: A natural number representing the number of samples taken from Sample$(\{u, v\})$ until u is the winner (last one not included).
$i = 0$
while True **do**
    winner = Sample$(\{u, v\})$
    if winner $= u$ **then**
        return i
    $i = i + 1$

Algorithm 10 `BalancedEstimateRatio`$(\mathcal{G}, i, j, A_1, A_2, \varepsilon, \alpha, \delta)$

Input: An (A_1, A_2, ε)-cluster graph $\mathcal{G} = (F, r, (C_1, \dots, C_T), (c_1, \dots, c_T))$, two natural numbers i and j representing the index of two clusters C_i and C_j in $\mathcal{G}$, parameters $A_1, A_2, \varepsilon, \alpha$, and δ.
Output: An estimate $r(c_i, c_j)$ of the ratio w_{c_i}/w_{c_j}.
$B_1 = \max\left\{\frac{2\varepsilon}{1-\varepsilon-\frac{3}{4}}, \frac{6}{(1-\varepsilon)}, \frac{24\varepsilon}{23-4\varepsilon}\right\}$ ▷ Note that $B_1 = O(1)$ for $\varepsilon \in (0, \frac{1}{5})$.
$N(\alpha, \varepsilon) = \frac{B_1^2}{\alpha\varepsilon^2}$
$M = \lceil 8 \log(2/\delta) \rceil$
$N = \left\lceil 2 \cdot A_1 \cdot \left(1 + \frac{A_1}{A_2}\right) N(\alpha, \varepsilon) \right\rceil$
$\xi = \left\lceil \frac{MN}{|C_j|} \right\rceil$
for $s \in C_j$ **do**
    for $\ell = 1, \dots, \xi$ **do**
        $X_{\ell,s} = r(c_j, s) \cdot$ `GetGeometric`(c_i, s)
Divide the first MN values of $\{X_{\ell,s}\}_{\ell,s}$ into M groups of size N: $\{X_1^{(1)}, \dots, X_N^{(1)}\}$, $\{X_1^{(2)}, \dots, X_N^{(2)}\}, \dots, \{X_1^{(M)}, \dots, X_N^{(M)}\}$.
for $\ell = 1, \dots, M$ **do**
    $Y_\ell = \frac{1}{N} \sum_{q=1}^{N} X_q^{(\ell)}$
$Y = \text{median}(\{Y_\ell\})$ ▷ Computes the median of the values $Y_1, \dots, Y_M$
if $Y \le \frac{3}{4} \cdot \alpha$ **then**
    return $r(c_i, c_j) = \infty$
return $r(c_i, c_j) = 1/Y$

In this section, we show a subroutine `BalancedEstimateRatio` that produces an estimate $r(c_i, c_j)$ of $\frac{w_{c_i}}{w_{c_j}}$ by spreading the queries among all items of the cluster C_j, instead of simply querying the pair (c_i, c_j) repeatedly.

The key observation behind this algorithm, is that one can obtain an unbiased estimator of the ratio $\frac{w_v}{w_u}$ by counting the number of Sample queries to $\{u, v\}$ needed before the oracle returns u as the winner (Lemma 36). Moreover, since the cluster graph produced in the first phase of the algorithm contains estimates of the ratios $\frac{w_u}{w_c}$ where c is the center of $C_{\gamma(u)}$, one can compose these estimates with estimates of ratios of the form $\frac{w_v}{w_u}$ to obtain estimators for $\frac{w_v}{w_c}$. In the end, the algorithm simply uses a median-of-means estimator to aggregate the result. This produces an accurate estimate by standard concentration results.

We first show the following.

Lemma 36. *Let Y be the output of* `GetGeometric`(u, v)*, then:*

$$\mathbb{E}[Y] = \frac{w_v}{w_u} \qquad \textit{and} \qquad \mathrm{Var}[Y] = \frac{\frac{w_v}{w_u+w_v}}{\left(\frac{w_u}{w_u+w_v}\right)^2} = \frac{w_v}{w_u}\left(1 + \frac{w_v}{w_u}\right).$$

Proof. The statement follows directly from the fact that $Y + 1 \sim \mathrm{Geom}\left(\frac{w_u}{w_u+w_v}\right)$. □

We will also make use of the following concentration bound:

Lemma 37. *Let $X_1, \ldots, X_N$ be independent r.v's, where $X_i \sim \mathrm{Ber}(\mu_i)$ and for each i, $\mu_i \geq 3/4$. Then, for $N \geq 8 \ln \frac{1}{\delta}$,*

$$\Pr\left[\sum_{i=1}^{N} X_i \leq \frac{N}{2}\right] \leq \delta.$$

Proof. Note that $\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] \geq \frac{3N}{4}$, thus, if $\sum_{i=1}^{N} X_i \leq \frac{N}{2}$ we also have $\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] - \sum_{i=1}^{N} X_i \geq \frac{N}{4}$. By Chernoff-Hoeffding inequality (see, e.g., (Dubhashi and Panconesi, 2009, Theorem 1.1)):

$$\Pr\left[\sum_{i=1}^{N} X_i \leq \frac{N}{2}\right] \leq \Pr\left[\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] - \sum_{i=1}^{N} X_i \geq \frac{N}{4}\right] \leq \exp\left(-2 \cdot \frac{N^2}{16} \cdot \frac{1}{N}\right) \leq \delta. \qquad \square$$

We then show the following.

Lemma 38. *Let $i, j \in [k]$ be such that $i > j$, and let $\mathcal{G} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be an (A_1, A_2, ε)-cluster graph for some $\varepsilon \in (0, 1/5)$. Then, with probability at least $1 - \delta$, the algorithm* `BalancedEstimateRatio`$(\mathcal{G}, i, j, A_1, A_2, \varepsilon, \alpha, \delta)$ *outputs $r(c_i, c_j) \in (0, \infty]$ such that:*

1. *If $\frac{w_{c_i}}{w_{c_j}} \leq \frac{1}{\alpha}$ then $r(c_i, c_j) \neq \infty$,*
2. *If $\frac{w_{c_i}}{w_{c_j}} \geq \frac{9}{\alpha}$ then $r(c_i, c_j) = \infty$,*
3. *If $r(c_i, c_j) \neq \infty$ then:*

$$r(c_i, c_j) \in (1 \pm 10\varepsilon)\frac{w_{c_i}}{w_{c_j}} \quad \text{and} \quad r(c_j, c_i) = \frac{1}{r(c_i, c_j)} \in (1 \pm 10\varepsilon)\frac{w_{c_j}}{w_{c_i}}.$$

Proof. Since $\mathcal{G}$ is a (A_1, A_2, ε)-cluster graph, we have, for every $s \in C_j$:

$$r(c_j, s) \in (1 \pm \varepsilon)\frac{w_{c_j}}{w_s}, \tag{30}$$

and:

$$\frac{w_{c_i}}{w_s} = \frac{w_{c_i}}{w_{c_j}} \cdot \frac{w_{c_j}}{w_s} \geq A_2 \cdot \frac{1}{A_1} = \frac{A_2}{A_1},$$

and hence:

$$\frac{w_{c_i}}{w_s + w_{c_i}} = \frac{1}{\frac{w_s}{w_{c_i}} + 1} \geq \frac{1}{\frac{A_1}{A_2} + 1} = \frac{A_2}{A_1 + A_2}. \tag{31}$$

By Lemma 36 and Equation (30), for any choice of $\ell \in [\xi]$ and any choice of $s \in C_j$:

$$\mathbb{E}\left[X_{\ell,s}\right] = r(c_j, s) \cdot \mathbb{E}\left[\texttt{GetGeometric}(c_i, s)\right] = r(c_j, s) \cdot \frac{w_s}{w_{c_i}} \in (1 \pm \varepsilon)\frac{w_{c_j}}{w_{c_i}},$$

and:

$$\begin{aligned}
\mathrm{Var}[X_{\ell,s}] &= \mathrm{Var}[r(c_j, s) \cdot \texttt{GetGeometric}(c_i, s)] \\
&= r(c_j, s)^2 \cdot \mathrm{Var}[\texttt{GetGeometric}(c_i, s)] \\
&= r(c_j, s)^2 \cdot \frac{w_s}{w_{c_i}} \cdot \left(1 + \frac{w_s}{w_{c_i}}\right)
\end{aligned}$$

$$
\begin{aligned}
&\leq (1+\varepsilon)^2 \cdot \left(\frac{w_{c_j}}{w_s}\right)^2 \cdot \frac{w_s}{w_{c_i}} \cdot \left(1+\frac{w_s}{w_{c_i}}\right) \\
&= (1+\varepsilon)^2 \cdot \frac{w_{c_j}}{w_s} \cdot \left(1+\frac{w_s}{w_{c_i}}\right) \cdot \frac{w_{c_j}}{w_{c_i}} \\
&\leq (1+\varepsilon)^2 \cdot A_1 \cdot \left(1+\frac{A_1}{A_2}\right) \cdot \frac{w_{c_j}}{w_{c_i}} \\
&\leq 2 \cdot A_1 \cdot \left(1+\frac{A_1}{A_2}\right) \cdot \frac{w_{c_j}}{w_{c_i}}.
\end{aligned}
$$

In particular consider Y_ℓ obtained as the average of $\{X_1^{(\ell)}, \ldots, X_N^{(\ell)}\}$, then:

$$
\mathbb{E}[Y_\ell] \in (1 \pm \varepsilon) \cdot \frac{w_{c_j}}{w_{c_i}} \tag{32}
$$

and by independence:

$$
\mathrm{Var}[Y_\ell] \leq \frac{1}{N} \max_{\substack{z \in [\xi] \\ s \in C_j}} \mathrm{Var}[X_{z,s}] \leq \frac{1}{N} \cdot 2 \cdot A_1 \cdot \left(1+\frac{A_1}{A_2}\right) \cdot \frac{w_{c_j}}{w_{c_i}} \leq \frac{\alpha \varepsilon^2}{B_1^2} \cdot \frac{w_{c_j}}{w_{c_i}},
$$

giving:

$$
2 \cdot \sigma(Y_\ell) = 2 \cdot \sqrt{\mathrm{Var}[Y_\ell]} \leq 2 \frac{\varepsilon}{B_1} \sqrt{\alpha \cdot \frac{w_{c_j}}{w_{c_i}}}.
$$

We are now ready to prove the three properties of the statement. We divide the proof depending on the value of $\frac{w_{c_i}}{w_{c_j}}$. First, if $\frac{w_{c_i}}{w_{c_j}} \geq \frac{9}{\alpha}$, then the algorithm can fail only if it returns a value different from ∞. Under the assumption that $\frac{w_{c_j}}{w_{c_i}} \leq \frac{\alpha}{9}$, we have:

$$
\mathbb{E}[Y_\ell] \leq (1+\varepsilon) \frac{w_{c_j}}{w_{c_i}} \leq \frac{1+\varepsilon}{9} \cdot \alpha
$$

$$
2\sigma(Y_\ell) \leq 2 \frac{\varepsilon}{B_1} \sqrt{\alpha \cdot \frac{w_{c_j}}{w_{c_i}}} \leq 2 \frac{\varepsilon}{B_1} \sqrt{\frac{\alpha^2}{9}} \leq \frac{2}{3} \cdot \frac{\varepsilon}{B_1} \cdot \alpha \leq \left(\frac{3}{4} - \frac{1+\varepsilon}{9}\right) \alpha
$$

By Chebyshev's Inequality:

$$
\begin{aligned}
\Pr\left[Y_\ell \geq \frac{3}{4} \cdot \alpha\right] &\leq \Pr\left[Y_\ell - \mathbb{E}[Y_\ell] \geq \frac{3}{4} \cdot \alpha - \mathbb{E}[Y_\ell]\right] \\
&\leq \Pr\left[Y_\ell - \mathbb{E}[Y_\ell] \geq \frac{3}{4} \cdot \alpha - \frac{(1+\varepsilon_1)}{9} \cdot \alpha\right] \\
&\leq \Pr\left[Y_\ell - \mathbb{E}[Y_\ell] \geq \left(\frac{3}{4} - \frac{(1+\varepsilon_1)}{9}\right) \cdot \alpha\right] \\
&\leq \Pr\left[Y_\ell - \mathbb{E}[Y_\ell] \geq 2 \cdot \sigma(Y_\ell)\right] \\
&\leq \frac{1}{4}.
\end{aligned}
$$

The algorithm only returns ∞ if the value Y, which the median of M Y_ℓ's, is smaller than $\frac{3}{4} \cdot \alpha$. This happens if and only if most of the Y_ℓ's are smaller than $\frac{3}{4} \cdot \alpha$. By Lemma 37 this happens with probability at least $1 - \delta$, as needed.

Suppose now that $\frac{1}{\alpha} < \frac{w_{c_i}}{w_{c_j}} \leq \frac{9}{\alpha}$. In this case the algorithm fails if it simultaneously returns a value different from ∞ and such value is not a good estimate for the ratios. We now show that if $\frac{w_{c_i}}{w_{c_j}} \leq \frac{9}{\alpha}$ then Y and $\frac{1}{Y}$ are good estimates with probability at least $1 - \frac{\delta}{2}$. Under the assumption that $\frac{w_{c_j}}{w_{c_i}} \geq \frac{\alpha}{9}$, we have:

$$2 \cdot \sigma(Y_\ell) \leq 2\frac{\varepsilon}{B_1}\sqrt{\alpha \cdot \frac{w_{c_j}}{w_{c_i}}} \leq 6 \cdot \frac{\varepsilon}{B_1} \cdot \frac{w_{c_j}}{w_{c_i}} \leq \varepsilon \cdot (1-\varepsilon) \cdot \frac{w_{c_j}}{w_{c_i}},$$

and hence, again by Chebyshev's Inequality:

$$\begin{aligned}
\Pr\left[Y_\ell \notin (1 \pm 3\varepsilon) \cdot \frac{w_{c_j}}{w_{c_i}}\right] &\leq \Pr\left[Y_\ell \notin (1 \pm \varepsilon)\mathbb{E}\left[Y_\ell\right]\right] \\
&= \Pr\left[|Y_\ell - \mathbb{E}\left[Y_\ell\right]| \geq \varepsilon \mathbb{E}\left[Y_\ell\right]\right] \\
&\leq \Pr\left[|Y_\ell - \mathbb{E}\left[Y_\ell\right]| \geq \varepsilon \cdot (1-\varepsilon) \cdot \frac{w_{c_j}}{w_{c_i}}\right] \\
&\leq \Pr\left[|Y_\ell - \mathbb{E}\left[Y_\ell\right]| \geq 2\sigma(Y_\ell)\right] \\
&\leq \frac{1}{4}.
\end{aligned}$$

where the first inequality follows from the fact that $(1 \pm 3\varepsilon)\frac{w_{c_j}}{w_{c_i}} \supseteq (1 \pm \varepsilon)\mathbb{E}\left[Y_\ell\right]$ by (32). Hence with probability at least $3/4$ each Y_ℓ lies in the range $(1 \pm 3\varepsilon)\frac{w_{c_j}}{w_{c_i}}$. Therefore, by the same argument as above, the median Y lies in the interval $(1 \pm 3\varepsilon)\frac{w_{c_j}}{w_{c_i}}$ with probability at least $1 - \frac{\delta}{2}$. If this holds then we also have:

$$(1 - 10\varepsilon) \cdot \frac{w_{c_i}}{w_{c_j}} \leq \frac{1}{1+3\varepsilon} \cdot \frac{w_{c_i}}{w_{c_j}} \leq \frac{1}{Y} \leq \frac{1}{1-3\varepsilon} \cdot \frac{w_{c_i}}{w_{c_j}} \leq (1 + 10\varepsilon) \cdot \frac{w_{c_i}}{w_{c_j}}.$$

Finally, suppose that $\frac{w_{c_i}}{w_{c_j}} \leq \frac{1}{\alpha}$. In this case the algorithm can fail if it returns ∞ or if it returns an inaccurate estimate. As shown above, the latter happens with probability at most $\frac{\delta}{2}$. Moreover, the algorithm will return a number different from ∞ with probability at least $1 - \frac{\delta}{2}$. Indeed, under the assumption $\frac{w_{c_j}}{w_{c_i}} \geq \alpha$ we have:

$$2 \cdot \sigma(Y_\ell) \leq 2\frac{\varepsilon}{B_1}\sqrt{\alpha\frac{w_{c_j}}{w_{c_i}}} \leq 2\frac{\varepsilon}{B_1}\sqrt{\left(\frac{w_{c_j}}{w_{c_i}}\right)^2} = 2\frac{\varepsilon}{B_1} \cdot \frac{w_{c_j}}{w_{c_i}} \leq \left(1 - \varepsilon - \frac{3}{4}\right)\frac{w_{c_j}}{w_{c_i}}.$$

Applying Chebyshev's inequality, we have:

$$\begin{aligned}
\Pr\left[Y_\ell \leq \frac{3}{4} \cdot \alpha\right] &= \Pr\left[\mathbb{E}\left[Y_\ell\right] - Y_\ell \geq \mathbb{E}\left[Y_\ell\right] - \frac{3}{4} \cdot \alpha\right] \\
&\leq \Pr\left[|\mathbb{E}\left[Y_\ell\right] - Y_\ell| \geq \mathbb{E}\left[Y_\ell\right] - \frac{3}{4} \cdot \alpha\right] \\
&\leq \Pr\left[|\mathbb{E}\left[Y_\ell\right] - Y_\ell| \geq (1-\varepsilon) \cdot \frac{w_{c_j}}{w_{c_i}} - \frac{3}{4} \cdot \alpha\right] \\
&\leq \Pr\left[|\mathbb{E}\left[Y_\ell\right] - Y_\ell| \geq \left(1 - \varepsilon - \frac{3}{4}\right) \cdot \frac{w_{c_j}}{w_{c_i}}\right]
\end{aligned}$$

$$\leq \Pr\left[\left|\mathbb{E}\left[Y_\ell\right] - Y_\ell\right| \geq 2\sigma(Y_\ell)\right] \leq \frac{1}{4}.$$

The algorithm returns ∞ only if most of the Y_ℓ's are smaller than $\frac{3}{4}\cdot\alpha$. By Lemma 37 this happens with probability at most $\frac{\delta}{2}$, as needed. A union bound concludes the proof. □

We now give a bound on the query complexity of `BalancedEstimateRatio`. We will use the following concentration bound for the sum of geometric random variables.

Lemma 39. *Let $\lambda \in \mathbb{R}_{>0}$ and $X_1, \ldots, X_n \sim \mathrm{Geom}(p)$ be independent, identically distributed geometric random variables with parameter p, then:*

$$\Pr\left[\sum_{i=1}^{n} X_i \geq \frac{n}{p} + \frac{\lambda}{p} + 1\right] \leq \exp\left(-\frac{2p\lambda^2}{n+\lambda+p}\right).$$

Proof. Let $\nu \in \mathbb{R}_{>0}$ such that $(\nu-1)p > n$. The probability that $\sum_{i=1}^{n} X_i \geq \nu$ is the probability that the sum of $\lceil\nu\rceil - 1$ independent Bernoulli random variables with parameter p is less than n. Formally, let $Y \sim \mathrm{Bin}(\lceil\nu\rceil - 1, p)$, we have:

$$\begin{aligned}\Pr\left[\sum_{i=1}^{n} X_i \geq \nu\right] &= \Pr\left[Y < n\right] = \Pr[Y - (\lceil\nu\rceil - 1)p < n - (\lceil\nu\rceil - 1)p] \\ &\leq \exp\left(-\frac{2(n-(\lceil\nu\rceil-1)p)^2}{\lceil\nu\rceil - 1}\right) \leq \exp\left(-\frac{2((\nu-1)p-n)^2}{\nu}\right),\end{aligned}$$

where the first inequality follows by Chernoff-Hoeffding inequality (see, e.g., (Dubhashi and Panconesi, 2009, Theorem 1.1)). Picking $\nu = \frac{n}{p} + \frac{\lambda}{p} + 1$ yields the result. □

Lemma 40. *For any choice of $\delta \in (0,1)$ and for $A_1 = \Theta(1)$, $A_2 = \Theta(1)$, a call to the algorithm* `BalancedEstimateRatio`$(\mathcal{G}, i, j, A_1, A_2, \varepsilon, \alpha, \delta)$*, with probability at least $1-\delta$ queries each pair $\{c_i, s\}$, with $s \in C_j$, at most $O\left(\left(1 + \frac{1}{\alpha\varepsilon^2|C_j|}\right)\log\frac{|C_j|}{\delta}\right)$ times.*

Proof. Note that `BalancedEstimateRatio`$(\mathcal{G}, i, j, A_1, A_2, \varepsilon_2, \delta)$ calls `GetGeometric`(c_i, s) for each $s \in C_j$, $\xi = O(MN/|C_j|)$ times. Let X_ℓ be the number of Sample queries made by the ℓth call to `GetGeometric`(c_i, s). Note that $X_\ell \sim \mathrm{Geom}(\frac{w_{c_i}}{w_s + w_{c_i}})$. By Equation (31) we have:

$$\frac{w_{c_i}}{w_s + w_{c_i}} \geq \frac{A_2}{A_1 + A_2},$$

and hence $\sum_{\ell=1}^{\xi} X_\ell$ is stochastically dominated by the sum of ξ independent identically distributed geometric random variables with parameter $p = \frac{A_2}{A_1+A_2}$.

By Lemma 39, we have that for $\lambda = \xi + \frac{2}{p}\ln\frac{10|C_j|}{\delta}$:

$$\begin{aligned}\Pr\left[\sum_{\ell=1}^{\xi} X_\ell \geq \frac{2\xi}{p} + \frac{2}{p^2}\ln\frac{10|C_j|}{\delta} + 1\right] &= \Pr\left[\sum_{\ell=1}^{\xi} X_\ell \geq \frac{\xi}{p} + \frac{\lambda}{p} + 1\right] \leq \exp\left(-\frac{2p\lambda^2}{\xi+\lambda+p}\right) \\ &\leq \exp\left(-\frac{2p\lambda^2}{3\lambda}\right) = \exp\left(-\frac{2p\lambda}{3}\right) \\ &\leq \exp\left(-\ln\frac{10|C_j|}{\delta}\right) = \frac{\delta}{10|C_j|}.\end{aligned}$$

Since $\frac{2\xi}{p} + \frac{2}{p^2}\ln\frac{10|C_j|}{\delta} + 1 = O\left(\left(1 + \frac{1}{\alpha\varepsilon^2|C_j|}\right)\log\frac{|C_j|}{\delta}\right)$ the lemma follows by the union bound. □

8.2 Computing an MNL: Obtaining an Estimation-Forest

We conclude this section by discussing how, given the subroutines described above, we can compute a (t, ε)-estimation-forest for an MNL M by making at most $O\left(\frac{\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)$ queries per pair.

We provide the pseudocode of our algorithm in Algorithm 11. Before proceeding, we give some intuition for the algorithm. Recall that our $O(n \log n)$ adaptive algorithm (Algorithm 4), starting from some center c_i, iteratively attempts to estimate the ratio of the weights of c_i and those of the previous centers $c_{i-1}, c_{i-2}, \dots$ until it observes an "∞". The first idea of this new algorithm is that we can limit ourselves to estimating the ratios between the weight of c_i and those of $\{c_{i-1}, \dots, c_{i-\Lambda(n)}\}$ for $\Lambda(n) = \Theta(\log n)$—this is because each time that we jump to a cluster with smaller cluster index the weights of the items decrease by at least a constant. The second idea is to choose a different threshold for establishing when the ratio of two items is infinite. In Algorithm 4 the threshold was decided so that if $r(c_i, c_j) = \infty$ then c_i alone wins with probability at least $1 - \varepsilon$ against all the items in $C_j \cup \dots \cup C_1$ put together. In this second algorithm, intuitively, we would like $r(c_i, c_j) = \infty$ if c_i alone wins against all the items in C_j with probability at least $1 - \frac{\varepsilon}{\Lambda(n)}$, as this is sufficient to obtain an $(t, O(\varepsilon))$-estimation-forest. This can be achieved by testing whether $\frac{w_{c_j}}{w_{c_i}} \leq \beta_j$ for $\beta_j = \Theta\left(\frac{\varepsilon}{|C_j|\cdot\Lambda(n)}\right)$. This new threshold will lead to overall more queries but it will allow us to query each pair at most $O(\log^2 n)$ many times.

A negative byproduct of this weaker thresholding is that we cannot stop estimating the ratios between c_i and the other centers as soon as we observe an infinity, but we should test c_i against all the items in $\{c_{i-1}, \dots, c_{i-\Lambda(n)}\}$. In turn, this might create cases where we observe $r(c_i, c_j) = \infty$ but still obtain a good estimate for the ratio of c_i and c_ℓ for $\ell > j$. To avoid this situation, we estimate the ratios between c_i and, in order, $c_{i-\Lambda(n)}, c_{i-\Lambda(n)+1}, \dots, c_{i-1}$. We call j_m the maximum index for which the estimate $r(c_i, c_{j_m})$ is not infinite—which is also the first one to occur. Now, for $\ell \in \{i-1, \dots, j_m+1\}$ we estimate the ratio between c_i and c_ℓ by using $r(c_i, c_{j_m})$ and $r(c_\ell, c_{j_m})$. We can then leverage the properties of the cluster graph to ensure that, with high probability, for all these choices of ℓ, $r(c_\ell, c_{j_m})$ will not be infinite.

We now analyze Algorithm 11. Our goal is to prove the following result.

Theorem 41. *For any choice of three numbers $\alpha, \varepsilon, \delta \in (0,1)$, with $\alpha = \Theta(1)$, with probability at least $1-\delta$, `BuildBalancedEstimationForest`$(\alpha, \varepsilon, \delta)$ makes at most $O\left(\frac{\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)$ queries to each pair of items and returns a $(5, \varepsilon)$-estimation-forest.*

Observe that, thanks to Theorem 29, the above result immediately implies Theorem 10. We will now prove Theorem 41 via a series of lemmas. We begin by analyzing the query complexity.

Lemma 42. *Let $\alpha, \varepsilon, \delta \in (0,1)$, with $\alpha = \Theta(1)$. Suppose that the call made to `QuicksortClustering` correctly computes a cluster graph. Then, with probability at least $1 - \frac{\delta}{2}$, we have that the algorithm `BuildBalancedEstimationForest`$(\alpha, \varepsilon, \delta)$ makes at most $O\left(\frac{\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)$ Sample queries on each pair of items, and it makes $O\left(\frac{n\log^2(n/\varepsilon)\log(n/\delta)}{\varepsilon^3}\right)$ queries in total.*

Proof. `QuicksortClustering` makes at most $O\left(\frac{\log(n/\delta)}{\varepsilon^2}\right)$ queries for each pair as stated in Proposition 35.

After that, all the Sample queries performed by `BuildBalancedEstimationForest` occur during a call to `BalancedEstimateRatio`.

Algorithm 11 `BuildBalancedEstimationForest`$(\alpha, \varepsilon, \delta)$

Input: Parameters $\alpha, \varepsilon, \delta \in (0,1)$, and access to a Sample oracle for an MNL M supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$.
Output: with probability $\geq 1-\delta$, a $(5,\varepsilon)$-estimation-forest
$\varepsilon_1 = \varepsilon/10$
$\varepsilon_2 = \varepsilon_1/30$
$(F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T)) = \texttt{QuicksortClustering}(\alpha, \varepsilon_2, \frac{\delta}{4})$
Let $\mathcal{G}$ be a copy of the cluster graph $(F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$
$\Lambda(n) = \lceil \log_{1/\alpha}(\frac{49 \cdot n}{\alpha \cdot \varepsilon_1}) \rceil$
for $i \in [T]$ **do**
    $\beta_i = \frac{\alpha^2 \cdot \varepsilon_1}{49 \cdot |C_i| \cdot \Lambda(n)}$
$i = T$
while $i > 1$ **do**
    $j_m = -1$
    $j = \max\{1, i - \Lambda(n)\}$
    while $j < i$ and $j_m = -1$ **do**
        $r(c_i, c_j) = \max\left\{\texttt{BalancedEstimateRatio}(\mathcal{G}, i, j, \frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon_2, \beta_j, \frac{\delta}{4n^2}), \frac{1}{\alpha^{i-j}}\right\}$
        if $r(c_i, c_j) < \infty$ **then**
            $j_m = j$
            $E = E \cup \{(c_i, c_j)\}$
        $j = j + 1$
    if $j_m = -1$ **then**
        // If all the estimates of $r(c_i, c_j)$ are ∞, the current tree ends here
        $i = i - 1$
    else
        // Otherwise, the values of $r(c_i, c_j)$ for $j \in \{i-1, ..., j_m+1\}$ is estimated using estimates for the ratios $w_{c_i}/w_{c_{j_m}}$ and $w_{c_{j_m}}/w_{c_j}$
        for $j = i-1, \ldots, j_m+1$ **do**
            $\rho = \texttt{BalancedEstimateRatio}(\mathcal{G}, j, j_m, \frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon_2, \frac{\beta_{j_m}}{9}, \frac{\delta}{4n^2})$
            if $\rho \in \{0, \infty\}$ **then**
                return failure
            $r(c_i, c_j) = \max\left\{\frac{r(c_i, c_{j_m})}{\rho}, \frac{1}{\alpha^{i-j}}\right\}$
            $E = E \cup \{(c_i, c_j)\}$
        $i = j_m$
return $(F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$

Note that, for each pair (i,j), `BalancedEstimateRatio`$(\mathcal{G},i,j,\dots)$ is called at most once. By Lemma 40, with probability $1-\frac{\delta}{4n^2}$, `BalancedEstimateRatio`$(\mathcal{G},i,j,\dots)$ makes at most

$$O\left(\left(1+\frac{1}{\beta_j\cdot|C_j|\cdot\varepsilon^2}\right)\log\left(\frac{n^2\cdot|C_j|}{\delta}\right)\right)=O\left(\frac{\Lambda(n)}{\varepsilon^3}\log\left(\frac{n}{\delta}\right)\right)=O\left(\frac{\log(\frac{n}{\varepsilon})\cdot\log\left(\frac{n}{\delta}\right)}{\varepsilon^3}\right)$$

queries to each pair (c_i,v), with $v\in C_j$.

Given that `BalancedEstimateRatio` gets called less than n^2 times, with probability at least $1-\frac{\delta}{4}\geq 1-\frac{\delta}{2}$, each pair (c_i,v) is queried at most $O\left(\frac{\log(n/\varepsilon)\log(n/\delta)}{\varepsilon^3}\right)$ times after the clustering algorithm.

Note also that for each j there are at most $\Lambda(n)$ values of i for which the algorithm makes a call to `BalancedEstimateRatio`$(\mathcal{G},i,j,\dots)$. Thus, the total number of queries is upper bounded by,

$$\sum_{j\in[T]}\Lambda(n)\cdot|C_j|\cdot O\left(\frac{\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)\leq O\left(\frac{n\log^2(n/\varepsilon)\log(n/\delta)}{\varepsilon^3}\right).\qquad\square$$

We now show that the algorithm computes a $(5,\varepsilon)$-estimation-forest. This consists in proving four properties. We will do so in a series of lemmas. We start by showing that short paths provide good estimates of the weights ratio.

Lemma 43. *Let $\varepsilon,\alpha,\delta\in(0,1)$, and suppose that each call to* `BalancedEstimateRatio` *as well as the call to* `QuicksortClustering` *is successful. Then,* `BuildBalancedEstimationForest`$(\alpha,\varepsilon,\delta)$ *does not return failure. Moreover, let $\mathcal{F}=(F=([n],E),r,(C_1,\dots,C_T),(c_1,\dots,c_T))$ be the output of the algorithm. Then, for $\varepsilon_1=\frac{\varepsilon}{10}$ and for any integer $t\geq 1$ and any $u,v\in[n]$ such that $d(u,v)\leq t$,*

$$(1-\varepsilon_1)^t\cdot\frac{w_u}{w_v}\leq r(P(u,v))\leq(1+\varepsilon_1)^t\cdot\frac{w_u}{w_v}.$$

In particular, if $d(u,v)\leq 5$, $r(P(u,v))\in(1\pm\varepsilon)\cdot\frac{w_u}{w_v}$.

Proof. Just as in to the proof of Lemma 24, it is sufficient to show that each edge (u,v) in the forest is associated with a $(1\pm\varepsilon_1)$ estimate of the ratio $\frac{w_u}{w_v}$. Since `QuicksortClustering` returned a $(\frac{7}{\alpha},\frac{1}{\alpha},\varepsilon_2)$-cluster graph with $\varepsilon_2=\frac{\varepsilon_1}{30}$, we have $r(c_i,v)\in(1\pm\varepsilon_2)\frac{w_{c_i}}{w_v}$ and $r(v,c_i)\in(1\pm\varepsilon_2)\frac{w_v}{w_{c_i}}$ for each edge (c_i,v) for $i\in[T]$, $v\in C_i$.

Consider now an edge (c_i,c_j), for $i>j$. This is either added in the internal **while** loop that looks for j_m or it is included in the internal **for** loop.

We first consider the **while** loop case. Since $i>j$ and $r(c_i,c_j)\neq\infty$, by Lemma 38, the call to `BalancedEstimateRatio` returned a value ζ such that $\zeta\in(1\pm10\varepsilon_2)\frac{w_{c_i}}{w_{c_j}}$ and $1/\zeta\in(1\pm10\varepsilon_2)\frac{w_{c_j}}{w_{c_i}}$. Moreover, by the properties of the $(\frac{7}{\alpha},\frac{1}{\alpha},\varepsilon_2)$-cluster graph, $\frac{w_{c_i}}{w_{c_j}}\geq\frac{1}{\alpha^{i-j}}$. This implies that $r(c_i,c_j)=\max\{\zeta,1/\alpha^{i-j}\}\in(1\pm10\varepsilon_2)\frac{w_{c_i}}{w_{c_j}}$ and $r(c_j,c_i)=1/r(c_i,c_j)=\min\{1/\zeta,\alpha^{i-j}\}\in(1\pm10\varepsilon_2)\frac{w_{c_j}}{w_{c_i}}$.

Consider now the **for** loop case. Note that we have $i>j>j_m$. Note that when the **for** loop is executed, the value of $r(c_i,c_{j_m})$ is never ∞. In particular the call made on Line 15 to `BalancedEstimateRatio`$(\mathcal{G},i,j_m,\frac{7}{\alpha},\frac{1}{\alpha},\varepsilon_2,\beta_{j_m},\frac{\delta}{4n^2})$ returned a value other than ∞, and hence, by Lemma 38, it must be true that $\frac{w_{c_i}}{w_{c_{j_m}}}\leq\frac{9}{\beta_{j_m}}$. Moreover, we have $w_{c_i}\geq w_{c_j}$, and thus, $\frac{w_{c_j}}{w_{c_{j_m}}}\leq\frac{9}{\beta_{j_m}}$. Then, by Lemma 38, the call made to `BalancedEstimateRatio`$(\mathcal{G},j,j_m,\frac{7}{\alpha},\frac{1}{\alpha},\varepsilon_2,\frac{\beta_{j_m}}{9},\frac{\delta}{4n^2})$ on

Line 26, must return $\rho \neq \infty$, and therefore $\rho \in (1 \pm 10\varepsilon_2)\frac{w_{c_j}}{w_{c_{j_m}}}$. Therefore, the algorithm does not return "failure" and moreover, we have,

$$(1-20\varepsilon_2)\frac{w_{c_i}}{w_{c_j}} \leq \frac{(1-10\varepsilon_2)}{(1+10\varepsilon_2)} \cdot \frac{w_{c_i}}{w_{c_j}} \leq \frac{r(c_i,c_{j_m})}{\rho} \leq \frac{(1+10\varepsilon_2)}{(1-10\varepsilon_2)} \cdot \frac{w_{c_i}}{w_{c_{j_m}}} \cdot \frac{w_{c_{j_m}}}{w_{c_j}} \leq (1+30\varepsilon_2)\frac{w_{c_i}}{w_{c_j}},$$

where we used that $\frac{1-a}{1+a} \geq 1-2a$ for $a \geq 0$ and $\frac{1+a}{1-a} \leq 1+3a$ for $a \in (0,1/3)$. Moreover, by the fact that $\frac{w_{c_i}}{w_{c_j}} \geq \frac{1}{\alpha^{i-j}}$ and since $\varepsilon_2 \leq \frac{\varepsilon_1}{30}$, we have $r(c_i,c_j) = \max\left\{\frac{r(c_i,c_{j_m})}{\rho}, \frac{1}{\alpha^{i-j}}\right\} \in (1 \pm \varepsilon_1)\frac{w_{c_i}}{w_{c_j}}$. One can similarly prove the result for $1/r(c_i,c_j)$. This directly concludes the proof. □

We now show that if two vertices are far away in the forest, then the ratio of their weights is negligible. We will do so by analyzing the structure of the tree similarly to the argument of Lemma 25. Figure 4 can be used as a reference to visualize the disposition of the vertices: although the tree in the figure is generated by Algorithm 4, the disposition of the vertices is similar in this proof.

Lemma 44. *Let $\varepsilon, \alpha, \delta \in (0,1)$, and suppose that each call to* `BalancedEstimateRatio` *as well as the call to* `QuicksortClustering` *is successful. Let $\mathcal{F} = (F = ([n], E), r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$ be the output of* `BuildBalancedEstimationForest`$(\alpha, \varepsilon, \delta)$*. Then, for any $u, v \in [n]$ such that $d(u,v) > 5$ and $\gamma(u) > \gamma(v)$, it holds that $\sum_{s \in H_v} \frac{w_s}{w_u} \leq \varepsilon$, where $H_v = \{s \in [n] \mid \gamma(s) \leq \gamma(v)\}$.*

Proof. We first consider the case where u and v are in the same connected component $\mathcal{C}$. Since $d(u,v) \geq 6$, $d(c_{\gamma(u)}, c_{\gamma(v)}) \geq 4$. Note that $\mathcal{C}$ is a tree. Let $R \in [T]$ be the maximum index such that $c_R \in \mathcal{C}$ and suppose the tree is rooted at c_R. Let c_z be the ancestor of $c_{\gamma(v)}$ at distance 2 from $c_{\gamma(v)}$. Note that c_z exists and by construction $\gamma(u) \geq z > \gamma(v)$. Thus, $w_{\gamma(u)} \geq w_{c_z}$. Since $c_{\gamma(v)}$ is not a child of c_z, for any $s \in H_v$, $c_{\gamma(s)}$ is not a child of c_z. By construction this means exactly one of two things must hold: either $z - \gamma(s) > \Lambda(n)$ or we observed `BalancedEstimateRatio`$(\mathcal{G}, z, \gamma(s), \ldots, \beta_{\gamma(s)}, \ldots) = \infty$ during the construction of the forest. Let us consider these two situations individually. Formally, let $A = \{s \in H_v \mid z - \gamma(s) > \Lambda(n)\}$ and $B = H_v \setminus A$. Consider any $b \in B$. We have, $1 \leq z - \gamma(b) \leq \Lambda(n)$. Moreover,

$$w_b \leq \frac{7}{\alpha} \cdot w_{c_{\gamma(b)}} \leq \frac{7}{\alpha} \cdot \beta_{\gamma(b)} \cdot w_{c_z} \leq \frac{\alpha \cdot \varepsilon_1}{7 \cdot |C_{\gamma(b)}| \cdot \Lambda(n)} \cdot w_{c_z}, \tag{33}$$

where the first inequality is by definition of $(\frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon_2)$-cluster graph, in the second inequality we use that, by Lemma 38, $\frac{w_{c_z}}{w_{c_{\gamma(b)}}} \geq \frac{1}{\beta_{\gamma(b)}}$, and the last inequality is by definition of $\beta_{\gamma(b)}$.

Let $K = \{\gamma(b) \mid b \in B\}$. By definition, $|K| \leq \Lambda(n)$, indeed, K can only contain values in $\{z-1, \ldots, z-\Lambda(n)\}$. Thus,

$$\sum_{b \in B} \frac{w_b}{w_{c_z}} \overset{(33)}{\leq} \sum_{b \in B} \frac{\alpha \cdot \varepsilon_1}{7 \cdot |C_{\gamma(b)}| \cdot \Lambda(n)} = \frac{\alpha\varepsilon_1}{7} \sum_{k \in K} \sum_{b \in C_k} \frac{1}{|C_k| \cdot \Lambda(n)} = \frac{\alpha\varepsilon_1}{7} \cdot \frac{|K| \cdot |C_k|}{|C_k| \cdot \Lambda(n)} \leq \frac{\alpha\varepsilon_1}{7}. \tag{34}$$

Consider now any $a \in A$. Since $z - \gamma(a) \geq \Lambda(n) + 1$, by the properties of $(\frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon_2)$-cluster graph we have that $w_{c_z} \geq \frac{1}{\alpha^{\Lambda(n)+1}} \cdot w_{c_{\gamma(a)}}$, and therefore:

$$w_a \leq \frac{7}{\alpha} \cdot w_{c_{\gamma(a)}} \leq 7\alpha^{\Lambda(n)} \cdot w_{c_z} \leq 7 \cdot \alpha^{\log_{1/\alpha}(\frac{49 \cdot n}{\alpha \cdot \varepsilon_1})} \cdot w_{c_z} \leq \frac{\alpha \cdot \varepsilon_1}{7n} \cdot w_{c_z}.$$

Thus,

$$\sum_{a \in A} \frac{w_a}{w_{c_z}} \leq \frac{|A|\alpha\varepsilon_1}{7n} \leq \frac{\alpha}{7} \cdot \varepsilon_1. \tag{35}$$

Note that, by the properties of the cluster graph and because $\gamma(u) \geq z$, we have that: $w_u \geq \frac{\alpha}{7} w_{c_{\gamma(u)}} \geq \frac{\alpha}{7} w_{c_z}$, and therefore,

$$\sum_{s \in H_v} \frac{w_s}{w_u} \leq \sum_{s \in H_v} \frac{7}{\alpha} \cdot \frac{w_s}{w_{c_z}} = \sum_{a \in A} \frac{7}{\alpha} \cdot \frac{w_a}{w_{c_z}} + \sum_{b \in B} \frac{7}{\alpha} \cdot \frac{w_b}{w_{c_z}} \overset{(34),(35)}{\leq} 2\varepsilon_1 \leq \varepsilon.$$

We now consider the case where u and v are in different connected components. Let z be the minimum index such that c_z is in the same connected component as u. By construction, $\gamma(u) \geq z > \gamma(v)$. Similarly to before, we can partition H_v into $A = \{s \in H_v \mid z - \gamma(s) > \Lambda(n)\}$ and $B = H_v \setminus A$. By construction, for each $b \in B$, we observed $\texttt{BalancedEstimateRatio}(\dots, z, \gamma(b), \dots, \beta_{\gamma(b)}, \dots) = \infty$. Thus, with an argument identical to before, we can prove $\frac{w_b}{w_{c_z}} \leq \frac{\alpha\varepsilon_1}{7|C_{\gamma(b)}|\Lambda(n)}$ for each $b \in B$ and $\frac{w_a}{w_{c_z}} \leq \frac{\alpha\varepsilon_1}{7n}$ for each $a \in A$, and this concludes the proof as before. □

We now show that even if we compute estimates along long paths, the estimates are still well-behaved. Again, Figure 4 can be used as a reference for the disposition of the vertices.

Lemma 45. *Let $\varepsilon, \alpha, \delta \in (0,1)$, and suppose that each call to* `BalancedEstimateRatio` *as well as the call to* `QuicksortClustering` *is successful. Let $\mathcal{F} = (F = ([n], E), r, (C_1, \dots, C_T), (c_1, \dots, c_T))$ be the output of* $\texttt{BuildBalancedEstimationForest}(\alpha, \varepsilon, \delta)$*. Suppose that $u, v \in [n]$ are in the same connected component $\mathcal{C}$ of F, and $\gamma(u) > \gamma(v)$ and $d(u,v) > 5$. Then, $\sum_{s \in K_v} r(P(s,u)) \leq \varepsilon$, where $K_v = \{s \in \mathcal{C} \mid \gamma(s) \leq \gamma(v)\}$.*

Proof. Since $d(u,v) \geq 6$, $d(c_{\gamma(u)}, c_{\gamma(v)}) \geq 4$. Note that $\mathcal{C}$ is a tree. Let $R \in [T]$ be the maximum index such that $c_R \in \mathcal{C}$ and suppose the tree is rooted at c_R. Let c_z be the ancestor of $c_{\gamma(v)}$ at distance 2 from $c_{\gamma(v)}$. Let c_x be a vertex such that, (i) $\gamma(u) \geq x \geq z$, (ii) $d(c_{\gamma(u)}, c_x) \leq 2$, and (iii) c_x is an ancestor of $c_{\gamma(v)}$. Note that there is always a sibling of $c_{\gamma(u)}$ with these properties (potentially, it might also be $c_x = c_{\gamma(u)}$ or $c_x = c_z$). Since $w_{c_{\gamma(u)}} \geq w_{c_x}$ and $d(u, c_x) \leq 3$, we have,

$$r(P(c_x, u)) \overset{\text{Lemma 43}}{\leq} (1+\varepsilon_1)^3 \frac{w_{c_x}}{w_u} \leq (1+\varepsilon_1)^3 \frac{w_{c_{\gamma(u)}}}{w_u} \leq \frac{7(1+\varepsilon_1)^3}{\alpha}, \tag{36}$$

where the last inequality follows by the properties of $(\frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon_2)$-cluster graph.

Note now that, if c_i is an ancestor of c_j for $i > j$, then, by construction, $r(P(c_i, c_j)) \geq \frac{1}{\alpha^{i-j}}$. Since c_x is an ancestor of c_z, we have:

$$r(P(c_z, c_x)) = 1/r(P(c_x, c_z)) \leq \alpha^{x-z}. \tag{37}$$

Let $K_v = \{s \in \mathcal{C} \mid \gamma(s) \leq \gamma(v)\}$. Let $A = \{s \in K_v \mid z - \gamma(s) > \Lambda(n)\}$ and $B = K_v \setminus A$. Note that c_z is an ancestor of any vertex in K_v. Consider any $a \in A$, by using the properties of the $(\frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon_2)$-cluster graph, we have

$$\begin{aligned} r(P(a, c_z)) = r(a, c_{\gamma(a)}) \cdot r(P(c_{\gamma(a)}, c_z)) &\leq (1+\varepsilon_2) \cdot \frac{w_a}{w_{c_{\gamma(a)}}} \cdot \alpha^{z-\gamma(a)} \leq 7(1+\varepsilon_2)\alpha^{z-\gamma(a)-1} \\ &\leq 7(1+\varepsilon_2)\alpha^{\Lambda(n)} \leq 7(1+\varepsilon_2)\alpha^{\log_{1/\alpha}(\frac{49 \cdot n}{\alpha \cdot \varepsilon_1})} \leq \frac{\alpha(1+\varepsilon_1)\varepsilon_1}{7n}. \end{aligned} \tag{38}$$

Then, we have,

$$r(P(a,u)) = r(P(a,c_z)) \cdot r(P(c_z,c_x)) \cdot r(P(c_x,u)) \overset{(37)}{\leq} r(P(a,c_z)) \cdot \alpha^{x-z} \cdot r(P(c_x,u))$$

$$\leq r(P(a,c_z)) \cdot r(P(c_x,u)) \overset{(36),(38)}{\leq} \frac{7\alpha(1+\varepsilon_1)^4\varepsilon_1}{7\alpha n} \leq \frac{2\varepsilon_1}{n}, \tag{39}$$

where we used that $\varepsilon_1 \leq \frac{1}{10}$, and thus $(1+\varepsilon_1)^4 < 2$.

Consider now any $b \in B$. Let c_y be the ancestor of $c_{\gamma(b)}$ at distance 2 from $c_{\gamma(b)}$. Note that c_y exists because $d(c_z, c_{\gamma(b)}) \geq 2$, and we have $z \geq y$ (it might also be $c_z = c_y$). In particular, c_y is still a descendant of c_z. Since $z - \gamma(b) \leq \Lambda(n)$, we also have $y - \gamma(b) \leq \Lambda(n)$. Since c_y is an ancestor of $c_{\gamma(b)}$, $d(c_y, c_{\gamma(b)}) = 2$ and $y - \gamma(b) \leq \Lambda(n)$, during the construction of the tree, it must have happened that $\texttt{BalancedEstimateRatio}(\dots, y, \gamma(b), \dots, \beta_{\gamma(b)}, \dots) = \infty$. But then, by Lemma 38,

$$\frac{w_{c_y}}{w_{c_{\gamma(b)}}} \geq \frac{1}{\beta_{\gamma(b)}}. \tag{40}$$

Putting these observations together, and also by using the definition of $(\frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon_2)$-cluster graph and Lemma 43, we obtain:

$$\begin{aligned} r(P(b,c_z)) &= r(P(b,c_{\gamma(b)})) \cdot r(c_{\gamma(b)}, c_y) \cdot r(c_y, c_z) \leq \frac{7(1+\varepsilon_2)}{\alpha} \cdot (1+\varepsilon_1)^2 \cdot \frac{w_{c_{\gamma(b)}}}{w_{c_y}} \cdot \alpha^{z-y} \\ &\overset{(40)}{\leq} 7(1+\varepsilon_1)^3 \cdot \beta_{\gamma(b)} \cdot \alpha^{z-y-1} \leq \frac{(1+\varepsilon_1)^3 \cdot \varepsilon_1 \cdot \alpha^{z-y+1}}{7 \cdot |C_{\gamma(b)}| \cdot \Lambda(n)} \leq \frac{\alpha \cdot (1+\varepsilon_1)^3 \cdot \varepsilon_1}{7 \cdot |C_{\gamma(b)}| \cdot \Lambda(n)}. \end{aligned} \tag{41}$$

Observe that, since $\varepsilon_1 \leq \frac{1}{10}$, it holds that $(1+\varepsilon_1)^6 < 2$, and we have that,

$$\begin{aligned} r(P(b,u)) = r(P(b,c_z)) \cdot r(P(c_z,c_x)) \cdot r(P(c_x,u)) &\overset{(41),(37),(36)}{\leq} \frac{\alpha \cdot (1+\varepsilon_1)^3 \cdot \varepsilon_1}{7 \cdot |C_{\gamma(b)}| \cdot \Lambda(n)} \cdot \alpha^{x-z} \cdot \frac{7(1+\varepsilon_1)^3}{\alpha} \\ &\leq \frac{(1+\varepsilon_1)^6 \varepsilon_1}{|C_{\gamma(b)}|\Lambda(n)} \leq \frac{2\varepsilon_1}{|C_{\gamma(b)}|\Lambda(n)}. \end{aligned} \tag{42}$$

Let $\mathcal{K} = \{\gamma(s) \mid s \in B\}$. Note that $|\mathcal{K}| \leq \Lambda(n)$. Finally, we have,

$$\begin{aligned} \sum_{s \in K_v} r(P(s,u)) = \sum_{a \in A} r(P(a,u)) + \sum_{b \in B} r(P(b,u)) &\overset{(39),(42)}{\leq} \frac{2|A|\varepsilon_1}{n} + \sum_{k \in \mathcal{K}} \sum_{b \in C_k} \frac{2\varepsilon_1}{|C_k|\Lambda(n)} \\ &\leq 2\varepsilon_1 + \frac{2|\mathcal{K}|\varepsilon_1}{\Lambda(n)} \leq 4\varepsilon_1 \leq \varepsilon. \end{aligned}$$

□

Proof of Theorem 41. With probability at least $1 - \frac{\delta}{4}$, `QuicksortClustering` correctly computes a cluster graph and queries each pair at most $O(\frac{\log(n/\delta)}{\varepsilon^2})$ times. Moreover, each call made to `BalancedEstimateRatio` is correct with probability at least $1 - \frac{\delta}{4n^2}$. We make no more than n^2 such calls, so they are all correct with probability at least $1 - \frac{\delta}{4}$. Thus, by the union bound, with probability at least $1 - \frac{\delta}{2}$, both the `QuicksortClustering` call and all the `BalancedEstimateRatio` calls are correct. Conditioning on this event, we have that Lemma 43, Lemma 44, and Lemma 45 hold and this gives the first three properties for a $(5, \varepsilon)$-estimation-forest except for the last part of the third point.

Note that vertices with the same cluster index are at distance at most two and for any tree in the forest, if we let x (resp. y) be the minimum (resp. maximum) cluster index in the tree, then the vertex set of the tree is $\cup_{i=x}^{y} C_i$. This ensures that all four properties are satisfied and therefore the algorithm returns a $(5, \varepsilon)$-estimation-forest. Moreover, by Lemma 42, each pair is queried at most $O(\frac{\log(n/\varepsilon)\log(n/\delta)}{\varepsilon^3})$ times with probability at least $1 - \frac{\delta}{2}$. Thus, with probability at least $1 - \delta$ the algorithm is both correct and has the desired query complexity. □

This in turn completes the proof of Theorem 10.

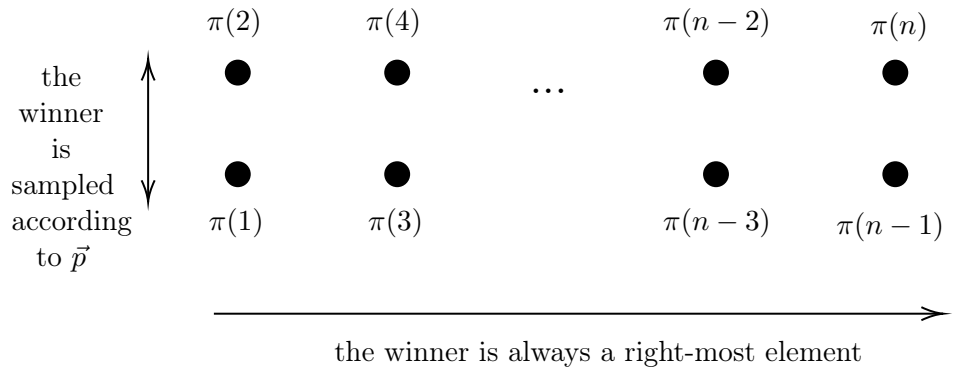


Figure 5: The instance $\overline{M}(n, \vec{p}, \pi)$ used for the lower bounds. Among two items in the same pair $\pi(2i-1)$ and $\pi(2i)$ the latter wins with probability $\vec{p}_i$. The winner is always an item of the right-most pair.

9 Lower Bounds

In order to prove lower bounds on the query complexity of the MNL learning task we consider the following family of instances.

We denote by $Sym(n)$ the set of permutations of $[n]$. For any even n, given a permutation $\pi \in Sym(n)$ we shall think of π as a way of partitioning the set $[n]$ into $n/2$ pairs $P_1^\pi, \dots, P_{n/2}^\pi$ where:

$$P_i^\pi := \{\pi(2i-1), \pi(2i)\}.$$

Given a non-empty set $S \subseteq [n]$ its *highest pair* with respect to π is the pair $P_{i(\pi,S)}^\pi$ where:

$$i(\pi, S) := \max_{P_i^\pi \cap S \neq \varnothing} i.$$

When π is the identity permutation, we simply write P_i and $i(S)$.

Definition 46 (Matching pseudo-MNL)**.** Given an even number $n \in \mathbb{N}$, a vector $\vec{p} \in [0,1]^{n/2}$, and a permutation $\pi \in Sym(n)$, the *matching pseudo-MNL* $\overline{M}(n, \vec{p}, \pi)$ supported on $[n]$ has the following Sample distributions. The winner of a slate S is always an item of its highest pair $P_{i(\pi,S)}^\pi = \{\pi(2 \cdot i(\pi, S) - 1), \pi(2 \cdot i(\pi, S))\}$. If only one of these items belongs to S then that item is the winner. Otherwise the winner is chosen to be $\pi(2 \cdot i(\pi, S))$ with probability $\vec{p}_{i(\pi,S)}$ and $\pi(2 \cdot i(\pi, S) - 1)$ with the remaining probability.

As a shorthand, we will denote by $\overline{M}(n, \vec{p}) := \overline{M}(n, \vec{p}, \mathrm{id})$ where $\mathrm{id} : [n] \to [n]$ is the identity permutation.

Note that the objects introduced in Definition 46 are not technically MNLs, but they are limits of a sequences of MNLs, and in particular, any lower bound on algorithms to learn objects of this kind immediately implies a lower bound on learning MNLs, as per the following result (which we prove in Appendix E).

Proposition 47. *Suppose there exists a potentially randomized algorithm $\mathcal{A}$ that takes as input $\varepsilon, \delta \in (0,1)$ and access to a* Sample *oracle for an MNL M supported on $[n]$, and after making at most $m(n, \varepsilon, \delta)$ queries outputs an MNL $\hat{M}$ such that:*

$$\Pr[d_\infty(M, \hat{M}) \leq \varepsilon] \geq 1 - \delta.$$

Then, the same algorithm, when given as input $\varepsilon, \delta \in (0,1)$ and access to a Sample *oracle for a matching pseudo-MNL $\overline{M}$ makes at most $m(n,\varepsilon,\delta)$ queries and outputs an MNL $\hat{M}$ such that:*

$$\Pr[d_\infty(\overline{M}, \hat{M}) \leq \varepsilon] \geq 1 - \delta.$$

An analogous result can be proved for d_{mul}, up to a factor of four in front of ε: any algorithm that learns any MNL within d_{mul} error of ε, also learns any matching pseudo-MNL within d_{mul} error of 4ε.

9.1 Lower Bound for Approximate Coin Selection

Definition 48 (Approximate top-$\frac{n}{2}$ coin selection problem)**.** Let $n \geq 2$ be even, $\varepsilon \in (0,1)$ and $\delta \in (0,1)$. The (ε,δ)-approximate top-$\frac{n}{2}$ coin selection problem is defined as follows. Given access to n Bernoulli distributions with unknown parameters $p_1, \ldots, p_n$, find, with probability at least $1-\delta$ a subset $C \subseteq [n]$, with $|C| = n/2$ such that:

$$\forall c \in C: \quad p_c \geq p_* - \varepsilon,$$

where p_* is the $(n/2)$th largest element of $\{p_1, \ldots, p_n\}$.

This problem is also known, in the multi-armed bandits literature, as the Explore-m problem or PAC subset selection in Stochastic Bandits (Kalyanakrishnan et al., 2012) for $m = n/2$. We can therefore leverage the lower bounds on this latter problem to obtain the following theorem.

Theorem 49 (Theorem 8 from Kalyanakrishnan et al. (2012)—Paraphrased)**.** *Let $n \geq 2$ be even, $\varepsilon \in \left(0, \sqrt{\frac{1}{32}}\right)$, $\delta \in \left(0, \frac{1}{4e}\right)$ then, for any (potentially randomized and adaptive) algorithm $\mathcal{A}$ for the (ε,δ)-approximate top-$\frac{n}{2}$ coin selection problem there is an instance on which the algorithm requires at least $\frac{1}{18375} \cdot \frac{n}{\varepsilon^2} \ln\left(\frac{n}{16\cdot\delta}\right)$ coin tosses.*

The previous result holds even for the expected number of coin tosses of the algorithm and moreover the hard instance is guaranteed to have biases in $\{\frac{1}{2}, \frac{1}{2} - 2\varepsilon, \frac{1}{2} + 2\varepsilon\}$ (see (Chaudhuri and Kalyanakrishnan, 2019, Theorem 3.1)). We also observe the following simple corollary, that we will use for our lower bound for d_{mul}.

Theorem 50. *Let $n \geq 2$ be even, $\eta \in \left(0, \sqrt{\frac{1}{32}}\right)$, $\rho \in (0,1]$, and $\delta \in \left(0, \frac{1}{4e}\right)$. Consider the $(\rho\eta,\delta)$-approximate top-$\frac{n}{2}$ coin selection problem restricted to instances with biases $q_j = \rho \cdot p_j$, where $p_j \in \{\frac{1}{2}, \frac{1}{2} - 2\eta, \frac{1}{2} + 2\eta\}$ for each $j \in [n]$. Any (possibly randomized and adaptive) algorithm that solves this problem with probability at least $1-\delta$ must make at least $\frac{1}{18375} \cdot \frac{n}{\rho\eta^2} \ln\left(\frac{n}{16\delta}\right)$ coin tosses in the worst case.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that solves the stated problem with at most T coin tosses in the worst case. We construct an algorithm $\mathcal{B}$ for the (η,δ)-approximate top-$n/2$ coin selection problem on the original biases $p_1, \ldots, p_n$ having expected sample complexity $\rho \cdot T$.

Algorithm $\mathcal{B}$ simulates $\mathcal{A}$. Whenever $\mathcal{A}$ requests a toss of coin j, algorithm $\mathcal{B}$ independently samples $b \sim \text{Ber}(\rho)$. If $b = 0$, it returns 0 to $\mathcal{A}$. Otherwise, it tosses the original coin j and returns its outcome. At the end, $\mathcal{B}$ returns the set identified by $\mathcal{A}$.

Each simulated toss is therefore distributed as $\text{Ber}(\rho \cdot p_j)$, exactly as required by $\mathcal{A}$. Therefore, with probability at least $1-\delta$, the set C returned satisfies, for each $c \in C$:

$$q_c \geq q_* - \rho\eta$$

$$\iff \rho \cdot p_c \geq \rho \cdot p_* - \rho\eta$$
$$\iff p_c \geq p_* - \eta.$$

Thus, $\mathcal{B}$ succeeds with probability at least $1-\delta$. By Theorem 49 (and Theorem 3.1 of Chaudhuri and Kalyanakrishnan (2019)), the expected number of coin tosses made by $\mathcal{B}$ on an instance where each bias is in the set $\{1/2, 1/2+2\eta, 1/2-2\eta\}$ must be at least: $\frac{1}{18375} \cdot \frac{n}{\eta^2} \ln\left(\frac{n}{16\delta}\right)$.

Since $\mathcal{A}$ makes at most T coin tosses, the expected number of coins that get actually tossed by $\mathcal{B}$ is $\rho \cdot T$. Therefore, we have:

$$\rho \cdot T \geq \frac{1}{18375} \cdot \frac{n}{\eta^2} \ln\left(\frac{n}{16\delta}\right)$$
$$\iff T \geq \frac{1}{18375} \cdot \frac{n}{\rho\eta^2} \ln\left(\frac{n}{16\delta}\right),$$

which concludes the proof. □

9.2 Lower Bounds for Adaptive Algorithms

In this section, we prove lower bounds for adaptive algorithms. We start with the following lower bound.

Theorem 12. *Any (possibly randomized and adaptive) algorithm that, given in input $\varepsilon, \delta \in (0, \frac{1}{32})$ and access to a* Sample *oracle for any MNL M, outputs an MNL $\hat{M}$ satisfying:*

$$\Pr[d_\infty(M, \hat{M}) \leq \varepsilon] \geq 1 - \delta,$$

must make $\Omega(\frac{n}{\varepsilon^2} \log \frac{n}{\delta})$ queries in the worst case.

The lower bound follows directly from the following lemma as well as Theorem 49.

Lemma 51. *Let $n \in \mathbb{N}$ be a multiple of 4, and let $\varepsilon \in (0, \frac{1}{2})$, $\delta \in (0,1)$. Suppose there is a (potentially randomized and adaptive) algorithm $\mathcal{A}$ that for any MNL M on $[n]$ makes at most $m(n, \varepsilon, \delta)$ queries and then outputs an MNL $\hat{M}$ which satisfies $d_\infty(M, \hat{M}) \leq \varepsilon$ with probability at least $1-\delta$. Then there exists an algorithm $\mathcal{B}$ for the $(2\varepsilon, \delta)$-approximate top-$\frac{n}{4}$ coin selection problem with worst-case query complexity $m(n, \varepsilon, \delta)$.*

Proof. Let $\vec{p} = p_1, \ldots, p_{n/2}$ be the unknown parameters of the $n/2$ Bernoulli distributions for an instance of the (ε, δ)-approximate top-$\frac{n}{4}$ coin selection problem. By Proposition 47, algorithm $\mathcal{A}$ must also satisfy $\Pr[d_\infty(\overline{M}(n, \vec{p}), \hat{M}) \leq \varepsilon] \geq 1 - \delta$, where $\overline{M}(n, \vec{p})$ is the matching pseudo-MNL of Definition 46. Moreover, $\mathcal{A}$ makes at most $m(n, \varepsilon, \delta)$ queries in this latter setting too.

We now describe an algorithm $\mathcal{B}$ that solves the instance of (ε, δ)-approximate top-$\frac{n}{4}$ coin selection with at most $m(n, \varepsilon, \delta)$ queries. Intuitively, $\mathcal{B}$ simulates algorithm $\mathcal{A}$ with access to $\overline{M}(n, \vec{p})$.

Let $P_{i(S)}$ be the highest pair in S. Whenever $\mathcal{A}$ makes a query $S_j \subseteq [n]$, if $|P_{i(S_j)} \cap S_j| = 1$ then $\mathcal{B}$ returns the unique item in $P_{i(S_j)} \cap S_j$; if instead $|P_{i(S_j)} \cap S_j| = 2$, then $\mathcal{B}$ samples from the $i(S_j)$th Bernoulli distribution (which has parameter $\vec{p}_{i(S_j)}$), and obtains a bit b_j and then it returns $2 \cdot i(S_j) - 1 + b_j$ to $\mathcal{A}$.

When $\mathcal{A}$ terminates, it outputs the weights of an MNL $\hat{M}$. $\mathcal{B}$ then sorts the elements of $[n/2]$ into a sequence $(s_1, \ldots, s_{n/2})$ so that:

$$\hat{M}_{\{2s_1-1, 2s_1\}}(2s_1) \geq \cdots \geq \hat{M}_{\{2s_{n/2}-1, 2s_{n/2}\}}(2s_{n/2})$$

and returns $\{s_1, \ldots, s_{n/4}\}$.

In order to see that this satisfies the required guarantees, we first note that, by construction, $\mathcal{B}$'s responses to $\mathcal{A}$'s queries are distributed like the responses of a Sample oracle for $\overline{M}(n, \vec{p})$. Therefore, under the conditioning that $d_\infty(\overline{M}(n, \vec{p}), \hat{M}) \leq \varepsilon$, we have:

$$\hat{M}_{\{2s_i-1,2s_i\}}(2s_i) \in \vec{p}_{s_i} \pm \varepsilon.$$

Hence, for all $i \in [n/4]$:

$$\vec{p}_{s_i} \geq \hat{M}_{\{2s_i-1,2s_i\}}(2s_i) - \varepsilon \geq \hat{M}_{\{2s_{n/4}-1,2s_{n/4}\}}(2s_{n/4}) - \varepsilon \geq \vec{p}_{s_{n/4}} - 2\varepsilon.$$

Since $\Pr[d_\infty(\overline{M}(n, \vec{p}), \hat{M}) \leq \varepsilon] \geq 1 - \delta$, we have that $\mathcal{B}$ solves the $(2\varepsilon, \delta)$-approximate top-$\frac{n}{4}$ coin selection problem with $m(n, \varepsilon, \delta)$ queries. □

For the d_{mul}-error we can prove a stronger lower bound in terms of ε. Specifically, we have the following result, whose proof is very similar to the previous one but we use Theorem 50 instead of Theorem 49 for our reduction.

Theorem 13. *Any (possibly randomized and adaptive) algorithm that, given in input $\varepsilon, \delta \in (0, \frac{1}{128})$ and access to a* Sample *oracle for any MNL M, outputs an MNL $\hat{M}$ satisfying:*

$$\Pr[d_{mul}(M, \hat{M}) \leq \varepsilon] \geq 1 - \delta,$$

must make $\Omega(\frac{n}{\varepsilon^3} \log \frac{n}{\delta})$ queries in the worst case.

Proof. Let $\mathcal{A}$ be an algorithm that with probability $1 - \delta$ learns any MNL within d_{mul} error of ε by making at most $m(n, \varepsilon, \delta)$ queries. By Proposition 47, the same algorithm must learn any matching pseudo-MNL within a d_{mul} error of 4ε by making the same number of queries (see also Section E).

Let n be a multiple of 4. Define $k = n/2$, $\rho = 16\varepsilon$, and $\eta = 8\varepsilon$. We construct an algorithm $\mathcal{B}$ that makes at most $m(n, \varepsilon, \delta)$ queries and solves the $(\rho\eta, \delta)$-approximate top-$k/2$ coin selection problem restricted to biases $q_i = \rho \cdot p_i$ where each p_i is in $\{1/2 - 2\eta, 1/2, 1/2 + 2\eta\}$. Note that, by Theorem 50, this is enough to conclude the proof. By our choice of parameters, $q_j \in \left\{8\varepsilon - 256\varepsilon^2,\ 8\varepsilon,\ 8\varepsilon + 256\varepsilon^2\right\}$. Since $\varepsilon \leq 1/128$, we have:

$$6\varepsilon \leq q_j \leq 10\varepsilon \qquad \forall j \in [k].$$

Let $\overline{M} = \overline{M}(n, \vec{q})$ be the matching pseudo-MNL of Definition 46, where item $2j$ has winning probability q_j in the pair $P_j = \{2j, 2j - 1\}$ for each $j \in [k]$. Algorithm $\mathcal{B}$ simulates $\mathcal{A}$ with oracle access to $\overline{M}$, exactly as in the proof of Lemma 51. By construction, this perfectly simulates the Sample oracle of $\overline{M}$, using at most one coin toss per query. Thus, $\mathcal{B}$ uses at most $m(n, \varepsilon, \delta)$ coin tosses. When $\mathcal{A}$ terminates and outputs an MNL $\hat{M}$, define

$$\hat{q}_j := \hat{M}_{P_j}(2j) \qquad \forall j \in [k].$$

Algorithm $\mathcal{B}$ sorts the coins in non-increasing order of $\hat{q}_j$ and returns the first $k/2$ coins.

We now prove that $\mathcal{B}$ solves $(\rho\eta, \delta)$-approximate top-$k/2$ selection problem with probability at least $1 - \delta$. Note that, with probability $1 - \delta$, we have:

$$d_{\text{mul}}(\overline{M}, \hat{M}) \leq 4\varepsilon.$$

Fix any $j \in [k]$ and consider the slate $P_j = \{2j, 2j-1\}$. Since $\overline{M}_{P_j}(2j), \overline{M}_{P_j}(2j-1) \geq 6\varepsilon$, we have that $\hat{M}$ must approximate multiplicatively both these winning probabilities (otherwise d_{mul} would be at least 6ε). Therefore, we have:

$$(1-4\varepsilon)q_j \leq \hat{q}_j \leq (1+4\varepsilon)q_j.$$

Since $q_j \leq 10\varepsilon$, we have:

$$|\hat{q}_j - q_j| \leq 4\varepsilon q_j \leq 40\varepsilon^2 \qquad \forall j \in [k].$$

Let q_* denote the $k/2$-th largest bias among $q_1, \ldots, q_k$. Since there are at least $k/2$ coins whose true bias is at least q_* and each of them is estimated within error $40\varepsilon^2$, we have that each returned item satisfies $\hat{q}_j \geq q_* - 40\varepsilon^2$. Therefore, for each returned item we have:

$$q_j \geq \hat{q}_j - 40\varepsilon^2 \geq q_* - 80\varepsilon^2 \geq q_* - \rho\eta,$$

which concludes the proof as $\mathcal{B}$ returns a solution to the $(\rho\eta, \delta)$-approximate top-$k/2$ selection problem. □

9.3 Lower Bounds for Non-Adaptive Algorithms

In this section, we prove the following:

Theorem 14. *Any (possibly randomized) non-adaptive algorithm that, given in input $\varepsilon \in (0, \frac{1}{32})$ and access to a* Sample *oracle for any MNL M, outputs an MNL $\hat{M}$ satisfying:*

$$\Pr[d_\infty(M, \hat{M}) \leq \varepsilon] \geq \frac{9}{10},$$

must make $\Omega(\frac{n^2}{\varepsilon^2} \log n)$ queries in the worst case.

We first introduce some key definitions. Note that every Sample query made by an algorithm can be identified with a subset $S_j \subseteq [n]$. We shall assume without loss of generality that all algorithms only make queries of cardinality at least 2. Consider an algorithm that has access to a Sample oracle for a matching pseudo-MNL $M(n, \vec{p}, \pi)$.

We say that query S_j *highlights* the pair P_i^π if P_i^π is the highest pair in S_j and $P_i^\pi \subseteq S_j$. We have the following result.

Lemma 52. *Let $\mathcal{A}$ be a potentially randomized non-adaptive MNL learning algorithm that makes at most m queries. Consider the process of running $\mathcal{A}$ with* Sample *access to the matching pseudo-MNL $\overline{M}(n, \vec{p}, \pi)$ of Definition 46, where $\pi \sim Sym(n)$ is a permutation of $[n]$ chosen uniformly at random, and $\vec{p} \in [0,1]^{n/2}$. Let Λ be the event that the number of queries made by the algorithm that highlight one of the pairs $P_1^\pi, \ldots, P_{\lfloor \frac{n}{12e} \rfloor}^\pi$ is less than or equal to $\frac{10m}{7n}$, then:*

$$\Pr[\Lambda] \geq \frac{9}{10}.$$

Proof. It is sufficient to show that the statement holds for any fixed, deterministic choice of the queries $S_1, \ldots, S_m$, as this will imply it holds for random queries, since π and $(S_1, \ldots, S_m)$ are independent of each other.

For every $i \in [n/2]$ let M_i be the random variable defined as:

$$M_i := |\{j \in [m] \mid S_j \text{ highlights } P_i^\pi\}|.$$

Let X_{ij} be the indicator random variable of the event that the query S_j highlights the pair P_i^π. Consider first any query S_j with $|S_j| > 2$ and $i \leq \frac{n}{12e}$. We have:

$$\begin{aligned}
\mathbb{E}[X_{ij}] &= \Pr_\pi[S_j \text{ highlights } P_i^\pi] \\
&= \frac{\binom{2i-2}{|S_j|-2}}{\binom{n}{|S_j|}} \\
&\leq \left(\frac{2i-2}{|S_j|-2}\right)^{|S_j|-2} \left(\frac{|S_j|}{n}\right)^{|S_j|} e^{|S_j|-2} \\
&= \left(\frac{2i-2}{|S_j|-2} \cdot \frac{|S_j|}{n}\right)^{|S_j|-2} \left(\frac{|S_j|}{n}\right)^{2} e^{|S_j|-2} \\
&\leq \left(3 \cdot \frac{2i-2}{n}\right)^{|S_j|-2} \left(\frac{|S_j|}{n}\right)^{2} e^{|S_j|-2} \\
&\leq \left(\frac{1}{2}\right)^{|S_j|-2} \left(\frac{|S_j|}{n}\right)^{2} && \left(\text{since } i \leq \frac{n}{12e}\right) \\
&\leq \frac{9}{2n^2},
\end{aligned}$$

where the first inequality follows by the fact that $\left(\frac{n}{k}\right)^k \leq \binom{n}{k}$ for $1 \leq k \leq n$ and $\binom{n}{k} \leq \left(\frac{n \cdot e}{k}\right)^k$ for $k \geq 1$, and the last step follows from the fact that the function $f(x) = x^2 2^{2-x}$ takes its maximum value over the positive integers at $x = 3$. Consider now any query S_j with $|S_j| = 2$ we have:

$$\mathbb{E}[X_{ij}] = \Pr_\pi[S_j \text{ highlights } P_i^\pi] = \frac{1}{\binom{n}{2}} = \frac{2}{n(n-1)} \leq \frac{9}{2n^2}.$$

where the last inequality holds for all $n \geq 2$. Summing up over all terms, we have:

$$\mathop{\mathbb{E}}_\pi\left[\sum_{i=1}^{\lfloor \frac{n}{12e} \rfloor} M_i\right] = \mathop{\mathbb{E}}_\pi\left[\sum_{i=1}^{\lfloor \frac{n}{12e} \rfloor} \sum_{j=1}^{m} X_{ij}\right] = \sum_{i=1}^{\lfloor \frac{n}{12e} \rfloor} \sum_{j=1}^{m} \mathbb{E}_\pi[X_{ij}] \leq \frac{9m}{24e \cdot n} \leq \frac{m}{7n}. \tag{43}$$

Let Λ be the event that $\sum_{i=1}^{\lfloor \frac{n}{12e} \rfloor} M_i \leq \frac{10m}{7n}$. Finally, by Markov's inequality and (43): $\Pr[\Lambda] \geq \frac{9}{10}$. $\square$

Lemma 53. *Consider any even $n \geq 150$, and let $\varepsilon \in (0, \frac{1}{2})$, $\delta \in (0, \frac{9}{10})$. Suppose there is a (potentially randomized) non-adaptive algorithm $\mathcal{A}$ that for any MNL M on $[n]$ makes at most $m(n, \varepsilon, \delta)$ non-adaptive queries and then outputs an MNL $\hat{M}$ which satisfies $d_\infty(M, \hat{M}) \leq \varepsilon$ with probability at least $1-\delta$. Then there exists an algorithm $\mathcal{B}$ for the $(2\varepsilon, \delta + \frac{1}{10})$-approximate top-$\frac{n_1}{2}$ coin selection problem with at most $\frac{10 \cdot m(n,\varepsilon,\delta)}{n}$ queries, where n_1 is the even number in $\{\lfloor \frac{n}{12e} \rfloor, \lfloor \frac{n}{12e} \rfloor - 1\}$.*

Proof. Algorithm $\mathcal{B}$ is given access to n_1 Bernoulli distributions with parameters $q_1, \ldots, q_{n_1}$ and needs to produce a subset of $\frac{n_1}{2}$ indices in $[n_1]$ corresponding to the approximate top-$\frac{n_1}{2}$ items. We construct $\mathcal{B}$ as follows. First, $\mathcal{B}$ samples a uniformly random permutation $\pi \in Sym(n)$. Then, it simulates $\mathcal{A}$ and obtains a set of queries, given as a multiset $S_1, \ldots, S_m$ of subsets of $[n]$. Then, it constructs a set $a_1, \ldots, a_m$ of responses to the queries as follows. If the query S_j highlights one of the pairs (the pair $P^\pi_{i(\pi,S_j)}$) and this pair is in $\{P_1^\pi, \ldots, P_{n_1}^\pi\}$, then $\mathcal{B}$ samples $x_j \in \{0,1\}$ from the $i(\pi, S_j)$th Bernoulli distribution (with parameter $q_{i(\pi,S_j)}$) and sets $a_j = \pi(2 \cdot i(\pi, S_j) - 1 + x_j)$. If the query S_j highlights another pair $P^\pi_{i(\pi,S_j)}$ where $i(\pi, S_j) > n_1$ then $\mathcal{B}$ samples $x_j \sim \{0,1\}$ uniformly

at random and returns $\pi(2 \cdot i(\pi, S_j) - i + x_j)$. Finally, if the query S_j does not highlight its highest pair with respect to π, then $\mathcal{B}$ sets a_j to the unique item in $S_j \cap P^{\pi}_{i(\pi,S_j)}$.

Then, $\mathcal{B}$ feeds the responses $a_1, \ldots, a_m$ back to $\mathcal{A}$, and $\mathcal{A}$ outputs an MNL $\hat{M}$. Finally, $\mathcal{B}$ sorts the elements of $[n_1]$ into a sequence $s_1, \ldots, s_{n_1}$ satisfying:

$$\hat{M}_{\{\pi(2s_1-1),\pi(2s_1)\}}(\pi(2s_1)) \geq \cdots \geq \hat{M}_{\{\pi(2s_{n_1}-1),\pi(2s_{n_1})\}}(\pi(2s_{n_1}))$$

and outputs $s_1, \ldots, s_{n_1/2}$.

We now prove that $\mathcal{B}$ is correct with high probability. First, observe that $\mathcal{B}$ is simulating access to Sample oracle for the matching pseudo-MNL $\overline{M}(n, \vec{p}, \pi)$, where π is chosen uniformly at random and:

$$\vec{p_i} := \begin{cases} q_i & \text{if } i \leq n_1 \\ \frac{1}{2} & \text{if } n_1 < i \leq \frac{n}{2}. \end{cases}$$

By Proposition 47, we have that, with probability at least $1 - \delta$, $d_{\infty}(\overline{M}(n, \vec{p}, \pi), \hat{M}) \leq \varepsilon$. If this event happens, $s_1, \ldots, s_{n_1/2}$ is a correct solution to the $(2\varepsilon, \delta)$-approximate top-$\frac{n_1}{2}$ coin selection instance—this can be proved with essentially the same argument as Lemma 51.

As it is written, the number of queries made by $\mathcal{B}$ to the Bernoulli distributions could be higher than $\frac{10m}{7n}$. Hence, in order to meet the requirements of the lemma, we modify $\mathcal{B}$ slightly. We introduce the following exception to the description above: if $\mathcal{B}$ ever needs to make more than $\frac{10m}{7n}$ queries to the Bernoulli distributions, it will instead output a uniformly random subset of $[n_1]$ of size $\frac{n_1}{2}$ and terminate. By Lemma 52 this happens with probability at most $\frac{1}{10}$, and the lemma follows. □

Theorem 14 then follows from Lemma 53 and Theorem 49.

10 Conclusions and Open Problems

In this paper, we considered the problem of learning an unknown MNL by making queries to a Sample oracle so that the learned weights can be used to provide an estimate to the distribution of each slate within an ℓ_1-error of ε. We developed two algorithms for this task: one for the adaptive setting and one for the non-adaptive setting.

Our adaptive algorithm has a query complexity of $O(\frac{n \log n}{\varepsilon^2})$ for $\delta = \frac{1}{\mathsf{poly}(n)}$, which is matched by our lower bound of $\Omega(\frac{n \log n}{\varepsilon^2})$. We also considered a stronger notion of learning, where the goal is to provide a multiplicative estimate to the distribution of each slate (except for sub-slates whose total probability is less than ε), and showed that for this stronger notion the optimal query complexity is $\Theta(\frac{n \log n}{\varepsilon^3})$.

Our non-adaptive algorithm has a query complexity of $O(\frac{n^2 \cdot \log n \cdot \log(n/\varepsilon)}{\varepsilon^3})$ nearly matching our $\Omega(\frac{n^2 \log n}{\varepsilon^2})$ non-adaptive lower bound. Closing this gap is a natural open problem.

Finally, our non-adaptive algorithm is based on an adaptive algorithm that queries each pair at most polylogarithmic many times. However, the latter is different from the two other adaptive algorithms we specifically designed for the adaptive setting. A possible direction for future work would be to find a single algorithm which can be used to match the query complexity of our algorithms in both the adaptive and non-adaptive setting. That is, an $O(n \log n)$ adaptive algorithm that queries each pair at most $\mathsf{polylog}(n)$ many times.

AI Usage Disclosure

All the results except for the $O(n \log n/\varepsilon^2)$ algorithm for d_1 (Section 7) and the $\Omega(n \log n/\varepsilon^3)$ lower bound for d_{mul} (Theorem 50 and Theorem 13) were obtained by the authors and were published in the conference version of this paper (Chierichetti et al., 2026). We obtained the other two results after extensive interactions with ChatGPT 5.6 Sol, which was given access to our conference paper and our attempted ideas. The final proofs were interpreted and written by the authors (with LLMs used for polishing the wording). The authors take full responsibility for all the work.

A Missing Proofs for Section 5

We first recall the following standard concentration result (see, e.g., Boucheron et al. (2013) Equation 2.10, page 36).

Theorem 54 (Bernstein's Inequality)**.** *Let $X_1, \ldots, X_N$ be i.i.d. r.v.'s in $[0,1]$ and each with mean μ and variance σ^2. Then, for $\lambda > 0$,*

$$\Pr\left[\frac{1}{N}\sum_{i=1}^{N} X_i - \mu \geq \lambda\right] \leq \exp\left(-\frac{\lambda^2 N}{2\sigma^2 + \frac{2}{3}\lambda}\right)$$
$$\Pr\left[\frac{1}{N}\sum_{i=1}^{N} X_i - \mu \leq -\lambda\right] \leq \exp\left(-\frac{\lambda^2 N}{2\sigma^2 + \frac{2}{3}\lambda}\right).$$

We will also use the following Chernoff-Bound (see, e.g., (Dubhashi and Panconesi, 2009, Theorem 1.1)):

Theorem 55 (Multiplicative Chernoff Bound)**.** *Let $X_1, \ldots, X_N$ be i.i.d. r.v.'s in $[0,1]$ and each with mean μ. Then, for $0 < \delta < 1$,*

$$\Pr\left[\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mu\right| \geq \delta \cdot \mu\right] \leq 2\exp\left(-\frac{\delta^2 \mu \cdot N}{3}\right).$$

Lemma 15 (`Compare` guarantees)**.** *For any $c, \varepsilon, \delta \in (0,1)$, $\texttt{Compare}(i, j, c, \varepsilon, \delta)$ makes $O\left(\frac{1}{c\varepsilon^2}\log\frac{1}{\delta}\right)$ queries and outputs a pair $(\hat{p}_i, \hat{p}_j)$ that, with probability at least $1-\delta$ satisfies, for $k \in \{i, j\}$:*

1. *If $M_{\{i,j\}}(k) \leq c/4$, then $\hat{p}_k = 0$,*
2. *If $M_{\{i,j\}}(k) \geq c$, then $\hat{p}_k \neq 0$,*
3. *If $\hat{p}_k \neq 0$ then $(1-\varepsilon)M_{\{i,j\}}(k) \leq \hat{p}_k \leq (1+\varepsilon)M_{\{i,j\}}(k)$.*

Proof. The bound on the number of queries follows directly from the pseudocode of the algorithm. We argue that the guarantees of the lemma hold with probability at least $1 - \frac{\delta}{2}$ for $\hat{p}_i$. By symmetry and the union bound, this implies that they hold for both $\hat{p}_i$ and $\hat{p}_j$ with probability at least $1-\delta$.

Let $p = M_{\{i,j\}}(i)$ and $X_1, \ldots, X_t \sim \text{Ber}(p)$ be the indicator random variables of the events that each call to $\mathsf{Sample}(\{i,j\})$ returns i, so that $\hat{p}_i = \frac{1}{m}\sum_{i=1}^{m} X_i$ and $\mathbb{E}[\hat{p}_i] = p$.

We divide the proof into three cases depending on the value of p, for all cases we prove that the guarantees hold with probability at least $1 - \frac{\delta}{2}$.

First, suppose that $p \leq c/4$. In this case, the algorithm can only fail if it returns $\hat{p}_i \neq 0$. By Bernstein's inequality, we find that the probability that $\hat{p}_i \neq 0$ is at most:

$$\Pr[\hat{p}_i \geq c/2] \leq \Pr[\hat{p}_i \geq p + c/4] \leq \exp\left(-\frac{m(c/4)^2}{2p(1-p) + \frac{c}{6}}\right) \leq \exp\left(-\frac{m(c/4)^2}{\frac{c}{2} + \frac{c}{6}}\right)$$

$$= \exp\left(-\frac{3 \cdot m \cdot c}{32}\right) \leq \frac{\delta}{6} \leq \frac{\delta}{2}.$$

Second, suppose $c/4 < p < c$. In this case, the algorithm can fail if it returns a value of $\hat{p}_i$ that simultaneously satisfies $\hat{p}_i \neq 0$ and $\hat{p}_i \notin (1 \pm \varepsilon)p$. We now show that, if $p \geq c/4$, then the probability that $\hat{p}_i \notin (1 \pm \varepsilon)p$ is at most $\delta/2$. By the multiplicative Chernoff bound (Theorem 55):

$$\Pr[|p - \hat{p}_i| \geq \varepsilon p] \leq 2e^{-\frac{\varepsilon^2 \cdot m \cdot p}{3}} \leq 2e^{-\frac{\varepsilon^2 \cdot m \cdot c}{12}} \leq \frac{\delta}{3} \leq \frac{\delta}{2}. \tag{44}$$

So if $c/4 < p < c$ the guarantees hold with probability at least $1 - \frac{\delta}{2}$.

Finally, consider the case $p \geq c$. In this case, the algorithm can fail either if $\hat{p}_i = 0$ or if $\hat{p}_i \notin (1 \pm \varepsilon)p$. By (44), the second event happens with probability at most $\delta/3$. By Bernstein's inequality, the first event happens with probability:

$$\Pr[\hat{p}_i \leq c/2] = \Pr[\hat{p}_i \leq p - (p - c/2)] \leq \exp\left(-\frac{m(p - c/2)^2}{2p(1-p) + \frac{2}{3} \cdot (p - c/2)}\right). \tag{45}$$

Here we consider two possibilities. Suppose that $c \leq p \leq 2c$. From (45), we obtain:

$$\Pr[\hat{p}_i \leq c/2] \leq \exp\left(-\frac{m(c/2)^2}{4c + c}\right) = \exp\left(-\frac{m \cdot c}{20}\right) \leq \frac{\delta}{6}.$$

If, instead, we have $p \geq 2c$, then $c/2 \leq p/4$. By (45), we have:

$$\begin{aligned}\Pr[\hat{p}_i \leq c/2] &\leq \exp\left(-\frac{m(p - p/4)^2}{2p(1-p) + \frac{2}{3}p}\right) \leq \exp\left(-\frac{m(\frac{3}{4} \cdot p)^2}{2p + p}\right) = \exp\left(-\frac{3 \cdot m \cdot p^2}{16 \cdot p}\right) \\ &= \exp\left(-\frac{3 \cdot m \cdot p}{16}\right) \leq \exp\left(-\frac{3 \cdot m \cdot c}{8}\right) \leq \frac{\delta}{6}.\end{aligned}$$

Thus, when $p \geq c$, the algorithm can fail with probability at most $\frac{\delta}{3} + \frac{\delta}{6} = \frac{\delta}{2}$. □

Lemma 16 (`EstimateRatio` Guarantees)**.** *Given two items i and j of $[n]$, and parameters α, ε, and δ in $(0, \frac{1}{2}]$, the algorithm* `EstimateRatio`$(i, j, \alpha, \varepsilon, \delta)$ *makes $O(\frac{1}{\alpha\varepsilon^2} \log \frac{1}{\delta})$ queries and produces an estimate $r(i, j)$ of the ratio $\frac{w_i}{w_j}$ that, with probability $1 - \delta$, satisfies the following guarantees:*

1. *If $\frac{w_i}{w_j} \leq \frac{\alpha}{3\alpha+4}$, then $r(i, j) = 0$.*
2. *If $\frac{w_i}{w_j} \geq \frac{3\alpha+4}{\alpha}$, then $r(i, j) = \infty$.*
3. *If $\frac{w_i}{w_j} \leq \frac{1}{\alpha}$, then $r(i, j) \neq \infty$, and if $\frac{w_i}{w_j} \geq \alpha$ then $r(i, j) \neq 0$.*
4. *Whenever $r(i, j) \notin \{0, \infty\}$:*

$$r(i, j) \in (1 \pm \varepsilon)\frac{w_i}{w_j} \quad \text{and} \quad \frac{1}{r(i, j)} \in (1 \pm \varepsilon)\frac{w_j}{w_i}.$$

Proof. We begin by noting that the query complexity bound for `EstimateRatio` follows directly from the query complexity of `Compare`. We now prove the rest of the guarantees. We will assume that $(\hat{p}_i, \hat{p}_j)$ satisfy the three guarantees in Lemma 15 and show that under this assumption, $r(i, j)$

and $1/r(i,j)$ satisfy the four conditions in the statement of this lemma. Since the former happens with probability at least $1-\delta$ (by Lemma 15), Lemma 16 will then follow.

If $\frac{w_i}{w_j} \leq \frac{\alpha}{3\alpha+4}$ then:

$$M_{\{i,j\}}(i) = \frac{w_i}{w_i+w_j} = \frac{1}{1+\frac{w_j}{w_i}} \leq \frac{1}{1+\frac{3\alpha+4}{\alpha}} = \frac{\alpha}{4(\alpha+1)} = \frac{c}{4},$$

and hence $\hat{p}_i = 0$, which implies $r(i,j) = 0$. If $\frac{w_i}{w_j} \geq \frac{3\alpha+4}{\alpha}$ then $\frac{w_j}{w_i} \leq \frac{\alpha}{3\alpha+4}$ and the same argument shows $r(i,j) = \infty$, yielding the first two conditions of this lemma.

If $\alpha \leq \frac{w_i}{w_j}$, then:

$$M_{\{i,j\}}(i) = \frac{w_i}{w_i+w_j} \geq \frac{w_i}{w_i+\frac{1}{\alpha}w_i} = \frac{\alpha}{\alpha+1} = c,$$

and hence $\hat{p}_i$ satisfies:

$$\hat{p}_i \in \left(1 \pm \frac{\varepsilon}{3}\right) \frac{w_i}{w_i+w_j}, \tag{46}$$

so that $\hat{p}_i \neq 0$ giving that $r(i,j) \neq 0$. Similarly if $\frac{w_i}{w_j} \leq \frac{1}{\alpha}$, then:

$$M_{\{i,j\}}(j) = \frac{w_j}{w_i+w_j} \geq \frac{w_j}{\frac{1}{\alpha}w_j+w_j} = \frac{\alpha}{\alpha+1} = c,$$

and hence we obtain estimates $\hat{p}_j$ of the winning probability of j in the slate $\{i,j\}$ satisfying:

$$\hat{p}_j \in \left(1 \pm \frac{\varepsilon}{3}\right) \frac{w_j}{w_i+w_j}. \tag{47}$$

so that $r(i,j) \neq \infty$. This implies condition 3.

Moreover, if the algorithm ever outputs $r(i,j)$ different from 0 and ∞, it must have been the case that both $\hat{p}_i$ and $\hat{p}_j$ were not zero. By Lemma 15 this implies $\hat{p}_i$ and $\hat{p}_j$ satisfy (46) and (47), and hence, $r(i,j) = \frac{\hat{p}_i}{\hat{p}_j}$ satisfies:

$$r(i,j) = \frac{\hat{p}_i}{\hat{p}_j} \leq \frac{(1+\frac{\varepsilon}{3})}{(1-\frac{\varepsilon}{3})} \cdot \frac{w_i}{w_j} \leq \left(1+3\cdot\frac{\varepsilon}{3}\right)\frac{w_i}{w_j} = (1+\varepsilon)\frac{w_i}{w_j}$$

and:

$$r(i,j) = \frac{\hat{p}_i}{\hat{p}_j} \geq \frac{(1-\frac{\varepsilon}{3})}{(1+\frac{\varepsilon}{3})} \cdot \frac{w_i}{w_j} \geq \left(1-2\cdot\frac{\varepsilon}{3}\right)\frac{w_i}{w_j} \geq (1-\varepsilon)\frac{w_i}{w_j},$$

where we used that $\frac{1+a}{1-a} \leq 1+3a$ and $\frac{1-a}{1+a} \geq 1-2a$ for $a \in (0,1/3)$. Similarly, $\frac{1}{r(i,j)} = \frac{\hat{p}_j}{\hat{p}_i}$ satisfies:

$$\frac{1}{r(i,j)} = \frac{\hat{p}_j}{\hat{p}_i} \leq \frac{(1+\frac{\varepsilon}{3})}{(1-\frac{\varepsilon}{3})} \cdot \frac{w_j}{w_i} \leq \left(1+3\cdot\frac{\varepsilon}{3}\right)\frac{w_j}{w_i} = (1+\varepsilon)\frac{w_j}{w_i}$$

and:

$$\frac{1}{r(i,j)} = \frac{\hat{p}_j}{\hat{p}_i} \geq \frac{(1-\frac{\varepsilon}{3})}{(1+\frac{\varepsilon}{3})} \cdot \frac{w_j}{w_i} \geq \left(1-2\cdot\frac{\varepsilon}{3}\right)\frac{w_j}{w_i} \geq (1-\varepsilon)\frac{w_j}{w_i},$$

yielding condition 4 above. □

B Missing Proofs for Section 6: Computing Approximate Orderings

In this section we prove the Theorem 18. Specifically, this is a simple corollary of a result of Falahatgar et al. (2018) which considered the following notion of ordering:

Definition 56 (Additive β-ordering)**.** An additive β-ordering for an MNL M supported on $[n]$ is an ordering $(s_1, \ldots, s_n)$ of the items of $[n]$ such that, for any pair i, j with $i < j$:

$$M_{\{s_i, s_j\}}(s_i) \leq \frac{1}{2} + \beta.$$

They showed that such an ordering can be computed efficiently. The following is an adaptation of their result, where we boost the success probability and make the running time explicit.

Theorem 57 (Adaptation of Theorem 9 of Falahatgar et al. (2018))**.** *Choose any $\beta \in (0, 1/2)$, $\delta \in (0, 1)$. There exists an algorithm that, with probability at least $1-\delta$, makes $O\left(\frac{n \cdot \log(n/\delta)}{\beta^2} \cdot \left(1 + \frac{\log(1/\delta)}{\log n}\right)\right)$ queries and returns an additive β-ordering. Moreover, the running time of the algorithm is proportional to the query complexity and the algorithm only queries pairs.*

Proof. Theorem 9 of Falahatgar et al. (2018) provides an algorithm `Binary-Search-Ranking`$([n], \beta)$ that, with probability at least $1 - 1/n$, makes at most $O(\frac{n \log n}{\beta^2})$ queries and returns an additive β-ordering.[6] Lemma 23 of Falahatgar et al. (2018) provides an algorithm `Rank-Check`(π, β, δ) such that, with error probability at most δ: (i) if π is an additive β-ordering of $[n]$ it returns true, (ii) if π is not an additive 3β-ordering of $[n]$ it returns false. If π is an additive β'-ordering, with $\beta' \in (\beta, 3\beta)$, `Rank-Check` can return either true or false. Moreover, `Rank-Check` always makes $O\left(\frac{n}{\beta^2} \log(\frac{n}{\delta})\right)$ queries.

We now use these two subroutines to boost the success probability of the algorithm. Let $\eta = \left\lceil \frac{\ln(2/\delta)}{\ln(n)} \right\rceil$. We run η independent instances of `Binary-Search-Ranking`$([n], \beta/3)$ in parallel. Whenever an instance terminates and outputs an ordering π, we run `Rank-Check`$(\pi, \frac{\beta}{3}, \frac{\delta}{2\eta})$ and if it returns true, we give π in output, otherwise we ignore π and continue running the other instances.

Observe that `Rank-Check` is run at most η times, and therefore, with probability at least $1 - \delta/2$ all the outputs given by this subroutine are correct. Observe also that, with probability at least $1 - \delta/2$, at least one of the η instances of `Binary-Search-Ranking` makes at most $O(\frac{n \log n}{\beta^2})$ queries and returns an additive $\beta/3$-ordering. When this happens, `Rank-Check` will surely return true. Moreover, if `Rank-Check` were to return true even before that, it means that the returned π must be an additive β'-ordering for $\beta' \in (\beta/3, \beta)$. Therefore, the output returned is correct. Moreover, the total query complexity is given by:

$$O\left(\eta \cdot \frac{n \log n}{\beta^2} + \eta \cdot \frac{n \log(\frac{n\eta}{\delta})}{\beta^2}\right) \leq O\left(\eta \cdot \frac{n \log(n/\delta)}{\beta^2}\right) \leq O\left(\frac{n \log(n/\delta)}{\beta^2} \cdot \left(1 + \frac{\log(1/\delta)}{\log n}\right)\right).$$

Regarding the running time, by inspecting the pseudocode, one can see that `Rank-Check` runs in time proportional to the query complexity. Algorithm `Binary-Search-Ranking` consists of several subroutines, some of which are provided in (Falahatgar et al., 2017). All but two points of `Binary-Search-Ranking` run in time proportional to the query complexity.

[6] To be precise, Falahatgar et al. (2018) showed this result in another model which generalizes MNLs (see, e.g., Appendix A of Yue et al. (2012)).

First, the subroutine `Build-Binary-Search-Tree` (Falahatgar et al., 2017) runs in time $O\left(\frac{n}{\log^3 n}\right)$ and therefore its running time can be charged to the query complexity. Specifically, note that, despite the fact that this subroutine is called multiple times in the pseudocode; the binary search tree generated is always the same, and hence it is sufficient to call it once. More precisely, their algorithm takes in input an integer $N = O\left(\frac{n}{\log^3 n}\right)$ and builds a complete binary tree T, where each vertex maintains three values (l, m, r), where it always holds $m = \lceil \frac{l+r}{2} \rceil$. The root starts with $l = 1$ and $r = N$, and the generic vertex (l, m, r), with $r - l > 1$, generates a left child with values $(l, \lceil \frac{l+m}{2} \rceil, m)$ and a right child with values $(m, \lceil \frac{m+r}{2} \rceil, r)$.

Second, line 4.b.(i) of the subroutine `Interval-Binary-Search` (Falahatgar et al., 2017) sorts an array of integers Q such that $|Q| = O(\log n)$. This instruction is repeated $O(n)$ times. If implemented as written, this would lead to an $O(n \log n \log \log n)$ runtime which is not chargeable to the query complexity. However, we can exploit how Q is constructed to make this step more efficient. Specifically, Q is constructed as follows: the algorithm starts on the root of T and it always moves to an adjacent vertex (either the parent or one of the two children); each time a vertex with values (l, m, r) is visited, the values l, m, r are added to Q. This random walk goes on for at most $O(\log n)$ times. We can therefore do the following: throughout the execution of this walk in T, we maintain a parallel binary search tree T' which is a copy of T but containing only the vertices visited during the random walk. Thus, T' can be constructed in time at most $O(|Q|)$. After the random walk, we can obtain a sorted array of the (unique) values in Q as follows: first put a value 1 (which will always be present in Q), then run an in-order visit of T' and output the midpoint m of each visited vertex, finally append a value N at the end (which will always be present in Q). Thus, this sorted array can be constructed in $O(|Q|)$. We note that outputting the sorted array without the values multiplicity (i.e., where each distinct value appears exactly once) is sufficient to correctly perform the other steps of the algorithm. However, for completeness, we remark that it would not be difficult to adapt the algorithm to work with multiplicity. In particular, we can do so as follows: in each vertex of T' we can save the multiplicity of the midpoint value m and two pointers pointing to the vertices that have the left (resp. right) value as their midpoint (except for values 1 and N which are dealt with separately)—all these pointers are easy to maintain. Then, each time the random walk moves to a new vertex, we need to update three counters, an operation which requires constant time. In summary, since this step takes time $O(|Q|)$, its time complexity can be charged to the query complexity of `Interval-Binary-Search`, and in general, it requires at most $O(n \log n)$-time overall.

By inspection one can also see that the algorithm only queries pairs. □

We can now prove the result for ε_o-orderings:

Theorem 18. *Let $\varepsilon_o, \delta \in (0, 1)$. There is an algorithm that given access to a* Sample *oracle for an MNL M supported on $[n]$, with probability at least $1-\delta$, makes $O\left(\frac{n \log(n/\delta)}{\varepsilon_o^2} \cdot \left(1 + \frac{\log(1/\delta)}{\log n}\right)\right)$ queries and returns an ε_o-ordering of the items of M. Moreover, all the queries made by the algorithm are to slates of size two and the algorithm runs in time proportional to the number of queries.*

Proof. Use Theorem 57 to compute an additive β-ordering $(s_1, \ldots, s_n)$ for $\beta = \frac{\varepsilon_o}{4}$. This ordering is also an ε_o-ordering. Indeed, for $i < j$,

$$\frac{w_{s_i}}{w_{s_i} + w_{s_j}} \leq \frac{1}{2} + \beta \iff w_{s_i} \leq \left(\frac{1}{2} + \beta\right)\left(w_{s_i} + w_{s_j}\right) \iff \left(\frac{1 - 2\beta}{1 + 2\beta}\right) w_{s_i} \leq w_{s_j}$$
$$\implies (1 - 4\beta) w_{s_i} \leq w_{s_j} \iff (1 - \varepsilon_o) w_{s_i} \leq w_{s_j},$$

where we used $1 - 4\beta \leq \frac{1-2\beta}{1+2\beta}$ for $\beta \in [0, 1]$. □

C Missing Proofs for Section 7

We will use the following concentration bound for binomial random variables (see, e.g., (Dubhashi and Panconesi, 2009, Theorem 1.1 and Theorem 1.2)):

Theorem 58 (Concentration of Binomial Random Variables)**.** *Let $X \sim \mathrm{Bin}(n,p)$ for $p \in (0,1)$. Then, for any $t > 0$, the following holds with probability at least $1 - 2e^{-t}$:*

$$\mathbb{E}[X] - \sqrt{2 \cdot \mathbb{E}[X] \cdot t} \leq X \leq \mathbb{E}[X] + \sqrt{2 \cdot \mathbb{E}[X] \cdot t} + \frac{2}{3} \cdot t.$$

Lemma 31 (`EstimateOneSidedRatio` Guarantees)**.** *Let $i, j \in [n]$ satisfy $w_i \geq w_j$. Let $Z \geq 1$, $\varepsilon, \delta \in (0,1)$. With probability at least $1 - \delta$, `EstimateOneSidedRatio`$(i, j, Z, \varepsilon, \delta)$ returns a value $r(i,j)$ satisfying the following guarantees:*

1. *If $r(i,j) \neq \infty$, then:*
$$\left(1 - \varepsilon \cdot \sqrt{\frac{w_i}{Z \cdot w_j}}\right) \frac{w_j}{w_i} \leq \frac{1}{r(i,j)} \leq \frac{w_j}{w_i}.$$
2. *If $r(i,j) = \infty$, then:*
$$\frac{w_j}{w_i} \leq \frac{\varepsilon^2}{2 \cdot Z}.$$

Moreover, the algorithm makes $O\left(\frac{Z \cdot \log(1/\delta)}{\varepsilon^2}\right)$ pairwise queries.

Proof. Define for convenience $p := w_j/(w_i + w_j)$ and $\mu = m \cdot p$. Since $w_i \geq w_j$, we have $p \leq 1/2$ and

$$\frac{w_j}{2w_i} \leq p \leq \frac{w_j}{w_i}. \tag{48}$$

Note that $X \sim \mathrm{Bin}(m,p)$ and $\mathbb{E}[X] = \mu$. By Theorem 58, choosing $t := L = \ln(2/\delta)$, we have, with probability at least $1 - 2e^{-L} = 1 - \delta$:

$$\mu - \sqrt{2\mu L} \leq X \leq \mu + \sqrt{2\mu L} + \frac{2L}{3}. \tag{49}$$

We condition on this event and distinguish the two possible outputs of the algorithm.

Case 1: $r(i,j) = \infty$**.** In this case, the algorithm observed $X < 64L$. Note that it must be $\mu < 128L$. Indeed, if it was $\mu \geq 128L$, then (49) would give: $X \geq \mu - \sqrt{2\mu L} > 64L$, which is a contradiction. By (48) and the choice of m, we have:

$$\frac{w_j}{w_i} \leq 2p = \frac{2\mu}{m} < \frac{256L}{m} \leq \frac{256 \cdot L \cdot \varepsilon^2}{512 \cdot Z \cdot L} = \frac{\varepsilon^2}{2Z},$$

which proves the second guarantee.

Case 2: $r(i,j) \neq \infty$**.** In this case, $X \geq 64L$. Note that $\mu \geq 32L$, otherwise by (49) we would have: $X \leq 32L + \sqrt{64L^2} + \frac{2}{3}L < 64L$.

Recall that the algorithm computes:

$$\hat{p} = \frac{X - 4\sqrt{XL}}{m}, \qquad r(i,j) = \frac{1 - \hat{p}}{\hat{p}}.$$

We start by showing that $\hat{p}$ is well defined, and in particular that $0 \leq \hat{p} \leq p \leq 1/2$.

Since $X \geq 64L$, we have $X - 4\sqrt{XL} \geq \frac{X}{2} > 0$, and hence $\hat{p} > 0$. Moreover, by $\mu \geq 32 \cdot L$ and (49) we have:

$$\frac{3\mu}{4} \leq X \leq 2\mu. \tag{50}$$

Note that:

$$X - \mu \overset{(49)}{\leq} \sqrt{2\mu L} + \frac{2L}{3} \overset{(50)}{\leq} \sqrt{\frac{8XL}{3}} + \frac{2L}{3} \overset{(X \geq 64L)}{\leq} \left(\sqrt{\frac{8}{3}} + \frac{1}{12}\right)\sqrt{XL} < 4\sqrt{XL}.$$

Therefore:

$$\hat{p} = \frac{X - 4\sqrt{XL}}{m} \leq \frac{\mu}{m} = p.$$

We now focus on the quantity $1 - \frac{w_i}{w_j \cdot r(i,j)}$. Note that since $\hat{p} \leq p$, we have:

$$\frac{w_i}{w_j \cdot r(i,j)} = \frac{\hat{p}(1-p)}{p(1-\hat{p})} \leq 1,$$

and therefore $1 - \frac{w_i}{w_j \cdot r(i,j)} \geq 0$. On the other hand:

$$\begin{aligned}
1 - \frac{w_i}{w_j \cdot r(i,j)} = 1 - \frac{\hat{p}(1-p)}{p(1-\hat{p})} = \frac{p - \hat{p}}{p(1-\hat{p})} &\overset{(\hat{p} \leq 1/2)}{\leq} 2 \cdot \frac{p - \hat{p}}{p} = 2 \cdot \frac{\mu - X + 4\sqrt{XL}}{m \cdot p} \\
&\overset{(49),(50)}{\leq} 2 \cdot \frac{\sqrt{2\mu L} + 4\sqrt{2\mu L}}{m \cdot p} \overset{(\mu = m \cdot p)}{\leq} 16 \cdot \sqrt{\frac{L}{m \cdot p}} \overset{(48)}{\leq} 16 \cdot \sqrt{\frac{L}{m} \cdot \frac{2 \cdot w_i}{w_j}} \\
&\overset{(\text{choice of } m)}{\leq} 16 \cdot \sqrt{\frac{\varepsilon^2}{256 \cdot Z} \cdot \frac{w_i}{w_j}} = \varepsilon \cdot \sqrt{\frac{w_i}{Z \cdot w_j}}.
\end{aligned}$$

Therefore:

$$0 \leq 1 - \frac{w_i}{w_j\, r(i,j)} \leq \varepsilon\sqrt{\frac{w_i}{Z w_j}}.$$

Rearranging the terms we obtain:

$$\left(1 - \varepsilon\sqrt{\frac{w_i}{Z w_j}}\right)\frac{w_j}{w_i} \leq \frac{1}{r(i,j)} \leq \frac{w_j}{w_i},$$

which concludes the proof. □

Lemma 32. *For any $x_1, \ldots, x_n \in [0,1]$ and $z \in \mathbb{R}$, we have:*

$$\left|1 - z \cdot \prod_{i=1}^{n} x_i\right| \leq |1 - z| + \sum_{i=1}^{n}(1 - x_i).$$

Proof. We have:

$$\begin{aligned}
\left|1 - z \cdot \prod_{i=1}^{n} x_i\right| &= \left|1 - \prod_{i=1}^{n} x_i + (1 - z) \cdot \prod_{i=1}^{n} x_i\right| \\
&\leq \left|1 - \prod_{i=1}^{n} x_i\right| + \left|(1 - z) \cdot \prod_{i=1}^{n} x_i\right|
\end{aligned}$$

$$\leq \sum_{i=1}^{n}(1-x_i) + |1-z|,$$

where we used that $1 - \prod_{i=1}^{n} x_i \leq \sum_{i=1}^{n}(1-x_i)$; indeed, if $n = 2$: $1 - x_1x_2 = 1 - x_1 + x_1(1-x_2) \leq (1-x_1) + (1-x_2)$, and for $n > 2$ we have:

$$1 - \prod_{i=1}^{n} x_i \leq \sum_{i=1}^{n-2}(1-x_i) + (1-x_{n-1}x_n) \leq \sum_{i=1}^{n}(1-x_i). \qquad \square$$

D Missing Proofs for Section 8

Proposition 35. *There exists an algorithm* `QuicksortClustering`$(\alpha, \varepsilon, \delta)$ *that, given parameters* $\alpha, \varepsilon, \delta \in (0,1)$ *and access to a* Sample *oracle for an MNL* M *supported on* $[n]$*, queries each pair of items at most* $O\left(\frac{\log(n/\delta)}{\alpha\varepsilon^2}\right)$ *times and that, with probability at least* $1-\delta$*, returns a* $(\frac{7}{\alpha}, \frac{1}{\alpha}, \varepsilon)$*-cluster graph.*

Proof. We describe the algorithm in simple steps. Starting from $S = [n]$, pick a uniform at random pivot $c \in S$. Compare the pivot c with all other items $s \in S \backslash \{c\}$ by calling `EstimateRatio`$(c, s, \alpha, \varepsilon, \frac{\delta}{n^2})$ and let $r(c,s)$ be the result of the comparison. Define three sets: $C = \{s \in S \backslash \{c\} \mid r(c,s) \notin \{0, \infty\}\}$, $R = \{s \in S \setminus \{c\} \mid r(c,s) = 0\}$ and $L = S \setminus (C \cup R)$. The set $C \cup \{c\}$ becomes a new cluster with center c. The algorithm then recurs on R and places the resulting clusters after $C \cup \{c\}$ and finally it recurs on L and places the resulting clusters before $C \cup \{c\}$.

Observe that, for each pair, the algorithm calls `EstimateRatio` at most once. Therefore, each pair is compared at most $O(\frac{\log(n/\delta)}{\alpha\varepsilon^2})$ times. Moreover the total number of calls to `EstimateRatio` is at most $O(|S|^2)$. Thus, by a union bound, all these calls are successful with probability at least $1-\delta$. We prove that this algorithm is correct conditioning on this event.

Let $(C_1, \ldots, C_T)$ and $(c_1, \ldots, c_T)$ be the resulting ordered clustering. Consider any c_i and $s \in C_i \setminus \{c_i\}$. Since $r(c_i, s) \notin \{0, \infty\}$, we have by Lemma 16 that $r(c_i, s) \in (1 \pm \varepsilon)\frac{w_{c_i}}{w_s}$ and $1/r(c_i, s) \in (1 \pm \varepsilon)\frac{w_s}{w_{c_i}}$. Moreover, $\frac{w_{c_i}}{w_s} \in (\frac{\alpha}{3\alpha+4}, \frac{3\alpha+4}{\alpha})$, and thus, $\frac{w_{c_i}}{w_s} \in (\frac{\alpha}{7}, \frac{7}{\alpha})$.

Consider now a center c. Consider any $v \in L$, since $r(c,v) = \infty$, by Lemma 16 it must be the case that $\frac{w_c}{w_v} \geq \frac{1}{\alpha}$. Similarly, for any $v \in R$, since $r(c,v) = 0$, by Lemma 16 it must be true that $\frac{w_c}{w_v} \leq \alpha$. Now observe that for centers c_i, c_{i+1}, it must either be that, during the execution of the algorithm, c_i was a pivot and c_{i+1} was in R or c_{i+1} was a pivot and c_i was in L. Thus, in either case, $\frac{w_{c_{i+1}}}{w_{c_i}} \geq \frac{1}{\alpha}$. $\square$

E Missing Proofs for Section 9: Extension to Pseudo-MNLs

We have observed that one of the issues encountered when learning MNLs is that the ratio of the weights of items might approach ∞. In this section, we show by a straight-forward limiting argument that any algorithm that approximately learns MNLs is also approximately learning objects that are not exactly MNLs, but behave exactly as if some of its weights were infinitely larger than others. This happens because these objects arise as the limits of sequences of MNLs.

We define a *subset distribution family* supported on $[n]$ as a collection of distributions $H = \{\nu_S\}_{S \in 2^{[n]} \setminus \{\varnothing\}}$, where each ν_S is a probability distribution over S and $2^{[n]}$ is the power set of $[n]$. A Sample oracle for a subset distribution family is defined as the oracle that on input $S \subseteq [n]$ with $S \neq \varnothing$ returns a sample from the distribution ν_S. Let $\mathcal{F}([n])$ be the collection of subset distribution

families supported on $[n]$. Let $\mathcal{M}([n]) \subseteq \mathcal{F}([n])$ be the set of MNLs supported on $[n]$ (where we identify an MNL with the collection of distributions it induces on the slates).

We call a function $d : \mathcal{F}(n) \times \mathcal{F}(n) \to [0,1]$ a C-distance metric if it is symmetric and it satisfies $d(F_1, F_3) \leq C \cdot (d(F_1, F_2) + d(F_2, F_3))$ for any $F_1, F_2, F_3 \in \mathcal{F}(n)$.

Define for convenience $d_{\text{sym}}(M, M') = \max\{d_{\text{mul}}(M, M'), d_{\text{mul}}(M', M)\}$. Note that d_{sym} does not respect the triangle inequality; however, it holds:

$$d_{\text{sym}}(M_1, M_3) \leq 2 \cdot (d_{\text{sym}}(M_1, M_2) + d_{\text{sym}}(M_2, M_3)) .$$

Note that $d_{\text{mul}}(M, M') \leq d_{\text{sym}}(M, M') \leq 2 \cdot d_{\text{mul}}(M, M')$, which implies that an algorithm that can learn within arbitrary accuracy in d_{mul} can do so also in d_{sym}.

In the rest of the section, we denote by $d(\cdot, \cdot)$ a C-distance metric on $\mathcal{F}([n])$. This can be taken to be one of $d_1(\cdot, \cdot)$, or $d_\infty(\cdot, \cdot)$ (with $C = 1$), or d_{sym} (with $C = 2$), and the results will apply in either case.

Definition 59 (Pseudo-MNL)**.** A pseudo-MNL $\overline{M} \in \mathcal{F}([n])$ is a limit of a sequence of MNLs with respect to d. That is, there exists a sequence $\{M^{(k)}\}_{k \in \mathbb{N}} \subseteq \mathcal{M}([n])$ such that

$$\lim_{k \to \infty} d(M^{(k)}, \overline{M}) = 0.$$

That is, a pseudo-MNL is a subset distribution family supported on $[n]$ that is the limit of a sequence of MNLs supported on $[n]$. Note that the class of pseudo-MNLs is the same independently whether d is d_1, d_∞ or d_{sym} as convergence in one distance implies convergence in the other two. A direct consequence of this definition is that every MNL is also a pseudo-MNL. From this observation it is clear that the task of learning pseudo-MNLs is no easier than that of learning MNLs, but as it turns out, it is no harder either. In fact, the key result we show is the following: any algorithm that solves the MNL learning problem, must also solve the problem of learning pseudo-MNLs.

Theorem 60. *For any $n \in \mathbb{N}$, $\delta \in (0,1)$, and $\varepsilon \in (0,1)$, let $\mathcal{A}$ be an algorithm that given access to a* Sample *oracle for any MNL M supported on $[n]$, makes at most $m(n, \varepsilon, \delta)$ queries and then returns an MNL $\hat{M}$ such that:*

$$\Pr[d(M, \hat{M}) \leq \varepsilon] \geq 1 - \delta.$$

Then, the same algorithm $\mathcal{A}$, when given access to a Sample *oracle for a pseudo-MNL $\overline{M}$ supported on $[n]$ makes at most $m(n, \varepsilon, \delta)$ queries and returns an MNL $\hat{M}$ such that:*

$$\Pr[d(\overline{M}, \hat{M}) \leq C \cdot \varepsilon] \geq 1 - \delta.$$

Proof. By definition of pseudo-MNL, there exists some sequence of MNLs $\{M^{(i)}\}_{i \in \mathbb{N}}$ such that:

$$\lim_{i \to \infty} d(M^{(i)}, \overline{M}) = 0. \tag{51}$$

(Recall here, that d is either d_1, d_∞, or d_{sym}). Without loss of generality, we assume that $\mathcal{A}$ is fully adaptive, that is, it queries a single subset S_j and receives a sample $a_j \in S_j$ before choosing its next query. An analogous argument shows that the result holds for algorithms that make batches of queries before obtaining answers to all the queries made in the batch (e.g., non-adaptive algorithms).

Let $\hat{M}$ be the random variable representing the MNL output by the algorithm $\mathcal{A}$. For any choice of i we consider the probability measure $\mathbb{P}_i$ assigning probabilities on events in the experiment in which $\mathcal{A}$ is run with access to a Sample oracle for $M^{(i)}$. We also consider the probability measure $\mathbb{P}_\infty$, which assigns probabilities to events based on the experiment in which $\mathcal{A}$ is run with access to a Sample oracle for $\overline{M}$.

Let $T = \{(S_j, a_j)\}_{j\in[m]}$ be the the sequence of query-response pairs produced by the interaction between the algorithm and the oracle (i.e., T is the *transcript*), where for brevity we use $m = m(n, \varepsilon, \delta)$. Note that T is random. Moreover, the number of possible transcripts is upper bounded by $f(n, m) = (2^{n-1} \cdot n)^m$, and since both n and m do not change with i, the cardinality of the set of possible transcripts is at most a constant with respect to i. Let:

$$\mathcal{T}_\infty = \{\tau \text{ is a transcript} \mid \mathbb{P}_\infty[T = \tau] > 0\}.$$

Observe that, since for each slate S and $s \in S$, $M_S^{(i)}(s) > 0$ for each i, we have that for any $\tau \in \mathcal{T}_\infty$ it holds that $\mathbb{P}_i[T = \tau] > 0$. Note that, since any transcript possible under $\overline{M}$ is also possible with any MNL $M^{(i)}$ and the algorithm has a worst-case complexity of m queries for MNLs, it must also make at most m queries when it interacts with $\overline{M}$.

Let $\mathcal{S} = \{(S, a) \mid \overline{M}_S(a) > 0\}$ and let $D = \min_{(S,a)\in\mathcal{S}}\{\overline{M}_S(a)\}$. Note that $D > 0$ and it might possibly depend on n. By Equation (51), there exists an N_0 such that for each $i > N_0$, $d(M^{(i)}, \overline{M}) \le \frac{D}{2}$. Therefore, for each $(S, a) \in \mathcal{S}$ and $i > N_0$:

$$M_S^{(i)}(a) \ge \overline{M}_S(a) - d(M^{(i)}, \overline{M}) \ge \frac{D}{2} \ge d(M^{(i)}, \overline{M}). \tag{52}$$

We will soon be interested in referring to specific parts of a transcript. To this end, we denote by $T_{\ell:k}$ the pairs $\{(S_j, a_j)\}_{j=\ell}^k$, by $T_j^q = S_j$ (the jth query in T) and by $T_j^a = a_j$ the jth answer or response in T. Fix a transcript $\tau = \{(S_j, a_j)\}_{j\in[m]} \in \mathcal{T}_\infty$. We have, for each $i > N_0$:

$$\begin{aligned}
\mathbb{P}_\infty[T = \tau] &= \prod_{j\in[m]} \mathbb{P}_\infty[T_j^a = a_j \mid T_{1:j-1} = \tau_{1:j-1} \wedge T_j^q = S_j] \cdot \mathbb{P}_\infty[T_j^q = S_j \mid T_{1:j-1} = \tau_{1:j-1}] \\
&= \prod_{j\in[m]} \overline{M}_{S_j}(a_j) \cdot \mathbb{P}_i[T_j^q = S_j \mid T_{1:j-1} = \tau_{1:j-1}] \\
&\ge \prod_{j\in[m]} \left(M_{S_j}^{(i)}(a_j) - d(M^{(i)}, \overline{M})\right) \cdot \mathbb{P}_i[T_j^q = S_j \mid T_{1:j-1} = \tau_{1:j-1}] \\
&= \prod_{j\in[m]} \left(\mathbb{P}_i[T_j^a = a_j \mid T_{1:j-1} = \tau_{1:j-1} \wedge T_j^q = S_j] - d(M^{(i)}, \overline{M})\right) \cdot \mathbb{P}_i[T_j^q = S_j \mid T_{1:j-1} = \tau_{1:j-1}] \\
&\ge \mathbb{P}_i[T = \tau] - 2^m \cdot d(M^{(i)}, \overline{M}),
\end{aligned}$$

where the first inequality is valid because by Equation (52) $M_{S_j}^{(i)}(a_j) - d(M^{(i)}, \overline{M}) \ge 0$. Consider now:

$$\mathcal{T}^{(i)} := \{\tau \text{ is a transcript} \mid \mathbb{P}_\infty[T = \tau] = 0, \mathbb{P}_i[T = \tau] > 0\}.$$

For any $\tau \in \mathcal{T}^{(i)}$, since $\mathbb{P}_\infty[T = \tau] = 0$ but $\mathbb{P}_i[T = \tau] > 0$, there must exists $(S_j, a_j) \in \tau$ such that $\overline{M}_{S_j}(a_j) = 0$. But then, $\mathbb{P}_i[T = \tau] \le M_{S_j}^{(i)}(a_j) \le d(M^{(i)}, \overline{M})$. Therefore, for any $\tau \in \mathcal{T}_\infty \cup \mathcal{T}^{(i)}$, we have, for each $i > N_0$:

$$\mathbb{P}_\infty[T = \tau] \ge \mathbb{P}_i[T = \tau] - 2^m \cdot d(M^{(i)}, \overline{M}). \tag{53}$$

Fix some $\varepsilon_1 > \varepsilon$. Since $\lim_{i\to\infty} d(M^{(i)}, \overline{M}) = 0$ there exists some N_1 such that for all $i > N_1$, we have $d(M^{(i)}, \overline{M}) < \varepsilon_1 - \varepsilon$. And hence, for all $i > \max\{N_0, N_1\}$:

$$\begin{aligned}
\mathbb{P}_\infty\left[d(\hat{M}, \overline{M}) \le C \cdot \varepsilon_1\right] &\ge \mathbb{P}_\infty\left[d(\hat{M}, M^{(i)}) + d(M^{(i)}, \overline{M}) \le \varepsilon_1\right] \\
&= \mathbb{P}_\infty\left[d(\hat{M}, M^{(i)}) \le \varepsilon_1 - d(M^{(i)}, \overline{M})\right]
\end{aligned}$$

$$
\begin{aligned}
&\geq \mathbb{P}_\infty\left[d(\hat{M}, M^{(i)}) \leq \varepsilon\right] \\
&= \sum_{\tau \in \mathcal{T}_\infty} \mathbb{P}_\infty\left[d(\hat{M}, M^{(i)}) \leq \varepsilon \,\middle|\, T = \tau\right] \mathbb{P}_\infty\left[T = \tau\right] \\
&= \sum_{\tau \in \mathcal{T}_\infty} \mathbb{P}_i\left[d(\hat{M}, M^{(i)}) \leq \varepsilon \,\middle|\, T = \tau\right] \mathbb{P}_\infty\left[T = \tau\right] \\
&= \sum_{\tau \in \mathcal{T}_\infty \cup \mathcal{T}^{(i)}} \mathbb{P}_i\left[d(\hat{M}, M^{(i)}) \leq \varepsilon \,\middle|\, T = \tau\right] \mathbb{P}_\infty\left[T = \tau\right] \\
&\overset{(53)}{\geq} \sum_{\tau \in \mathcal{T}_\infty \cup \mathcal{T}^{(i)}} \mathbb{P}_i\left[d(\hat{M}, M^{(i)}) \leq \varepsilon \,\middle|\, T = \tau\right] \left(\mathbb{P}_i\left[T = \tau\right] - 2^m \cdot d(M^{(i)}, \overline{M})\right) \\
&\geq \mathbb{P}_i\left[d(\hat{M}, M^{(i)}) \leq \varepsilon\right] - |\mathcal{T}_\infty \cup \mathcal{T}^{(i)}| \cdot 2^m \cdot d(M^{(i)}, \overline{M}) \\
&\geq 1 - \delta - |\mathcal{T}_\infty \cup \mathcal{T}^{(i)}| \cdot 2^m \cdot d(M^{(i)}, \overline{M}).
\end{aligned}
$$

The above holds for all choices of $i > \max\{N_0, N_1\}$. Note that $|\mathcal{T}_\infty \cup \mathcal{T}^{(i)}|$ can be upper bounded by the total number of valid transcripts which is upper bounded by $f(n, m)$ and does not depend on i. Then, by taking the limit $i \to \infty$, and by (51) and the fact that m does not depend on i we have:

$$\mathbb{P}_\infty\left[d(\hat{M}, \overline{M}) \leq C \cdot \varepsilon_1\right] \geq 1 - \delta.$$

This holds for all $\varepsilon_1 > \varepsilon$. Define the sequence $\{\varepsilon^{(i)}\}_{i \in \mathbb{N}}$, by:

$$\varepsilon^{(i)} := \varepsilon + \frac{1}{i},$$

and for every $i \in \mathbb{N}$, let $\mathcal{E}_i$ be the event that $d(\hat{M}, \overline{M}) \in (C \cdot \varepsilon, C \cdot \varepsilon^{(i)}]$. By the union bound, we have, for all $i \in \mathbb{N}$,

$$\mathbb{P}_\infty[d(\hat{M}, \overline{M}) \leq C \cdot \varepsilon] \geq \mathbb{P}_\infty[d(\hat{M}, \overline{M}) \leq C \cdot \varepsilon^{(i)}] - \mathbb{P}_\infty[\mathcal{E}_i] \geq 1 - \delta - \mathbb{P}_\infty[\mathcal{E}_i],$$

where the second step follows from the derivation above. Note that $\forall i \in \mathbb{N}$, $\mathcal{E}_{i+1} \subseteq \mathcal{E}_i$, and that $\bigcap_{i \in \mathbb{N}} \mathcal{E}_i = \varnothing$. Hence, by taking the limit we obtain:

$$\mathbb{P}_\infty[d(\hat{M}, \overline{M}) \leq C \cdot \varepsilon] \geq 1 - \delta - \lim_{i \to \infty} \mathbb{P}_\infty[\mathcal{E}_i] = 1 - \delta - \mathbb{P}_\infty[\varnothing] = 1 - \delta. \qquad \square$$

Pseudo-MNLs have a very specific structure: they can be partitioned into an ordered sequence of disjoint MNLs so that the winner is always an item of the first MNL that intersects the queried slate, and the probability of winning among the items with this property is proportional to their weight in their respective MNL.

For concreteness, we now outline an example of pseudo-MNL. In Section 9 we introduced the following definition.

Definition 46 (Matching pseudo-MNL)**.** Given an even number $n \in \mathbb{N}$, a vector $\vec{p} \in [0,1]^{n/2}$, and a permutation $\pi \in Sym(n)$, the *matching pseudo-MNL* $\overline{M}(n, \vec{p}, \pi)$ supported on $[n]$ has the following Sample distributions. The winner of a slate S is always an item of its highest pair $P^\pi_{i(\pi,S)} = \{\pi(2 \cdot i(\pi, S) - 1), \pi(2 \cdot i(\pi, S))\}$. If only one of these items belongs to S then that item is the winner. Otherwise the winner is chosen to be $\pi(2 \cdot i(\pi, S))$ with probability $\vec{p}_{i(\pi,S)}$ and $\pi(2 \cdot i(\pi, S) - 1)$ with the remaining probability.

Where we defined:

$$i(S,\pi) := \max_{i:P_i^\pi \cap S \neq \varnothing} i$$

as the index of the highest pair $P_1^\pi, \ldots, P_{n/2}^\pi$ that intersects with the slate S.

In Section 9, we make use of Proposition 47 which states that that any algorithm that can learn MNLs, must also learn the matching pseudo-MNLs of Definition 46. We now show that matching pseudo-MNLs are indeed pseudo-MNLs. Therefore, Proposition 47 is an immediate corollary of Theorem 60 and of Lemma 61 below.

Lemma 61. *For any even $n \in \mathbb{N}$, $\vec{p} \in [0,1]^{n/2}$ and $\pi \in Sym(n)$, $\overline{M}(n,\vec{p},\pi)$ is a pseudo-MNL.*

Proof. For simplicity we denote by $\overline{M} := \overline{M}(n,\vec{p},\pi)$ We show that there is a sequence $\{M^{(j)}\}_{j\in\mathbb{N}}$ of MNLs with the property that for all $\varepsilon > 0$ there exists some $j_\varepsilon \in \mathbb{N}$ such that for all $j \geq j_\varepsilon$ it holds that $d_\infty(M^{(j)}, \overline{M}) \leq \varepsilon$. Note that this in turns implies that the same is true if d_∞ is replaced by d_1, by simply picking a value of ε that is n times smaller.

For each j we define the MNL $M^{(j)}$ as the MNL induced by the weights $w_1^{(j)}, \ldots, w_n^{(j)}$, defined in the following iterative way:

$$w_{\pi(1)}^{(j)} = 1,$$

and, for every odd $k \in [n]$, $k \geq 3$:

$$w_{\pi(k)}^{(j)} = j^3 \cdot w_{\pi(k-1)}^{(j)}$$

and for every even $k \in [n]$:

$$w_{\pi(k)}^{(j)} = \begin{cases} w_{\pi(k-1)}^{(j)} \cdot \frac{\vec{p}_{k/2}}{1-\vec{p}_{k/2}} & \text{if } \vec{p}_{k/2} \notin \{0,1\}, \\ \frac{w_{\pi(k-1)}^{(j)}}{j} & \text{if } \vec{p}_{k/2} = 0, \\ w_{\pi(k-1)}^{(j)} \cdot j & \text{if } \vec{p}_{k/2} = 1. \end{cases} \tag{54}$$

We now show that, for every choice of $\varepsilon > 0$ for all sufficiently large j, we have that for every slate S:

$$\|M_S^{(j)} - \overline{M}_S\|_\infty \leq \varepsilon.$$

In particular, given ε, we choose $j \in \mathbb{N}$ such that:

$$j \geq \frac{2n}{\varepsilon}. \tag{55}$$

Fix a slate S, and let $i^* = i(S,\pi)$. By construction, we have that for any $v \in P_{i^*}^\pi$, and $u \in S \setminus P_{i^*}^\pi$:

$$\frac{w_v^{(j)}}{w_u^{(j)}} \geq j. \tag{56}$$

Suppose $|S \cap P_{i^*}^\pi| = 1$. Then the distribution of winners over S for $\overline{M}(n,\vec{p},\pi)$ is given by:

$$\forall u \in S: \quad \overline{M}_S(u) = \begin{cases} 1 & \text{if } u \in S \cap P_{i^*}^\pi, \\ 0 & \text{otherwise.} \end{cases}$$

In this case, if $u \in S \cap P_{i^*}^\pi$:

$$1 \geq M_S^{(j)}(u) = \frac{w_u^{(j)}}{\sum_{\ell\in S} w_\ell^{(j)}} = \frac{1}{1 + \sum_{\ell \in S\setminus\{u\}} \frac{w_\ell^{(j)}}{w_u^{(j)}}} = \frac{1}{1 + \sum_{\ell \in S\setminus P_{i^*}^\pi} \frac{w_\ell^{(j)}}{w_u^{(j)}}} \overset{(56)}{\geq} \frac{j}{j+n} \overset{(55)}{\geq} 1-\varepsilon,$$

giving:

$$\left|M_S^{(j)}(u) - \overline{M}_S(u)\right| \leq \varepsilon,$$

while if $u \in S \setminus P_{i^*}^{\pi}$, let v be the unique item in $S \cap P_{i^*}^{\pi}$, we have:

$$0 \leq M_S^{(j)}(u) = \frac{w_u^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}} \leq \frac{w_u^{(j)}}{w_v^{(j)}} \overset{(56)}{\leq} \frac{1}{j} \overset{(55)}{\leq} \varepsilon, \tag{57}$$

and:

$$|M_S^{(j)}(u) - \overline{M}_S(u)| \leq \varepsilon,$$

giving:

$$\|M_S^{(j)} - \overline{M}_S\|_\infty \leq \varepsilon,$$

as needed.

Suppose now that $|S \cap P_{i^*}^{\pi}| = 2$. In this case, the distribution of winners over S for $\overline{M}(n, \vec{p}, \pi)$ is given by:

$$\overline{M}_S(u) = \begin{cases} 0 & \text{if } i \notin P_{i^*}^{\pi}, \\ \vec{p}_{i^*} & \text{if } u \in P_{i^*}^{\pi} \text{ and } \pi^{-1}(u) \text{ is even}, \\ 1 - \vec{p}_{i^*} & \text{if } u \in P_{i^*}^{\pi} \text{ and } \pi^{-1}(u) \text{ is odd}, \end{cases}$$

for every $u \in S$.

We now show that for all $u \in S$, $|M_S^{(j)}(u) - \overline{M}_S(u)| \leq \varepsilon$. We divide the proof into three cases: (Case 1:) $u \notin P_{i^*}^{\pi}$, (Case 2:) $u \in P_{i^*}^{\pi}$ and $\pi^{-1}(u)$ is even, and (Case 3:) $u \in P_{i^*}^{\pi}$ and $\pi^{-1}(u)$ is odd.

Case 1. If $u \notin P_{i^*}^{\pi}$, then, by Equation (57), $|M_S^{(j)}(u) - \overline{M}_S(u)| = M_S^{(j)}(u) \leq \varepsilon$.

Case 2. If $u \in P_{i^*}^{\pi}$ and $\pi^{-1}(u)$ is even, let v be the unique item in $P_{i^*}^{\pi}$ such that $v \neq u$. If $\vec{p}_{i^*} = 0$, we have $\overline{M}_S(u) = 0$, while:

$$0 \leq M_S^{(j)}(u) = \frac{w_u^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}} \leq \frac{w_u^{(j)}}{w_v^{(j)}} \overset{(54)}{\leq} \frac{1}{j} \overset{(55)}{\leq} \varepsilon$$

and hence:

$$|\overline{M}_S(u) - M_S^{(j)}(u)| \leq \varepsilon.$$

On the other hand, if $\vec{p}_{i^*} = 1$, we have $\overline{M}_S(u) = 1$, and:

$$1 \geq M_S^{(j)}(u) = \frac{w_u^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}} \overset{(54),(56)}{\geq} \frac{j}{j+n} \overset{(55)}{\geq} 1 - \varepsilon$$

and hence:

$$|\overline{M}_S(u) - M_S^{(j)}(u)| \leq \varepsilon.$$

Finally, if $\vec{p}_{i^*} \notin \{0, 1\}$ we have $\overline{M}_S(u) = \vec{p}_{i^*}$ and:

$$M_S^{(j)}(u) = \frac{w_u^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}} = \frac{w_u^{(j)}}{w_u^{(j)} + w_v^{(j)}} \cdot \frac{w_u^{(j)} + w_v^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}}.$$

We also have:

$$1 \geq \frac{w_u^{(j)} + w_v^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}} = \frac{w_u^{(j)} + w_v^{(j)}}{w_u^{(j)} + w_v^{(j)} + \sum_{\ell \in S \setminus P_{i^*}^\pi} w_\ell^{(j)}} = \frac{1}{1 + \frac{\sum_{\ell \in S \setminus P_{i^*}^\pi} w_\ell^{(j)}}{w_u^{(j)} + w_v^{(j)}}} \overset{(56)}{\geq} \frac{j}{j+n} \overset{(55)}{\geq} 1 - \varepsilon. \quad (58)$$

Hence:

$$\vec{p}_{i^*} = \frac{w_u^{(j)}}{w_u^{(j)} + w_v^{(j)}} \geq M_S^{(j)}(u) = \frac{w_u^{(j)}}{w_u^{(j)} + w_v^{(j)}} \cdot \frac{w_u^{(j)} + w_v^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}} \geq \vec{p}_{i^*} \cdot (1 - \varepsilon). \quad (59)$$

Giving:

$$|\overline{M}_S(u) - M_S^{(j)}(u)| \leq \varepsilon \vec{p}_{i^*} \leq \varepsilon.$$

Case 3. If $u \in P_{i^*}^\pi$ and $\pi^{-1}(u)$ is odd, let v be the unique item in $P_{i^*}^\pi$ such that $v \neq u$. We have $\overline{M}_S(u) = 1 - \vec{p}_{i^*}$. Note that $\pi^{-1}(v)$ is even, thus, by using (58) and (59) we get:

$$M_S^{(j)}(u) = M_S^{(j)}(u) + M_S^{(j)}(v) - M_S^{(j)}(v) = \frac{w_u^{(j)} + w_v^{(j)}}{\sum_{\ell \in S} w_\ell^{(j)}} - M_S^{(j)}(v) \in (1 - \vec{p}_{i^*}) \pm \varepsilon,$$

and hence:

$$|\overline{M}_S(u) - M_S^{(j)}(u)| \leq \varepsilon.$$

This then gives $\|\overline{M}_S - M_S^{(j)}\|_\infty \leq \varepsilon$ as needed. □

F An Additive Approximation for Pairs is not Sufficient

In the next theorem we prove that, for any constant $\varepsilon \in (0, 1/9)$, if one has an additive α-approximation to the distribution for all the slates of size 2, then, one must have $\alpha \leq \frac{9\varepsilon}{n}$ in order for this to guarantee an error of ε on all slates.

Theorem 62. *For any $\varepsilon \in (0, 1)$, there exist two families of MNLs $\{M_1^{(n)}\}_{n \in \mathbb{N}}$ and $\{M_2^{(n)}\}_{n \in \mathbb{N}}$ such that, for each n:*

1. *$M_1^{(n)}$ and $M_2^{(n)}$ are supported on $[n]$,*
2. *For every pair $u, v \in [n]$:*
$$|M_{1,\{u,v\}}^{(n)}(u) - M_{2,\{u,v\}}^{(n)}(u)| \leq \frac{\varepsilon}{n},$$
3. *$d_\infty(M_1^{(n)}, M_2^{(n)}) \geq \frac{\varepsilon}{9}$.*

Proof. Consider the following families of MNLs: in $M_1^{(n)}$ there are $n-1$ items of weight 1 and 1 item (Item 1) of weight n. In $M_2^{(n)}$ there are $n-1$ items of weight $1+\varepsilon$ and one item (Item 1) of weight n.

Condition (1) above is satisfied by construction. We now verify Condition (2). For any pair u, v where both u and v are not 1, we have $M_{1,\{u,v\}}^{(n)}(u) = M_{2,\{u,v\}}^{(n)}(u)$. On the other hand, if one of u and v is equal to 1, we can assume without loss of generality that $v = 1$, we then have:

$$M_{1,\{u,v\}}^{(n)}(u) = \frac{1}{n+1},$$

and:

$$M^{(n)}_{2,\{u,v\}}(u) = \frac{1+\varepsilon}{n+1+\varepsilon} \geq \frac{1}{n+1}.$$

On the other hand:

$$M^{(n)}_{2,\{u,v\}}(u) = \frac{1}{n+1+\varepsilon} + \frac{\varepsilon}{n+1+\varepsilon} \leq \frac{1}{n+1} + \frac{\varepsilon}{n}$$

and hence:

$$|M^{(n)}_{1,\{u,v\}}(u) - M^{(n)}_{2,\{u,v\}}(u)| \leq \frac{\varepsilon}{n},$$

as needed. Finally, we verify Condition (3). Consider the full slate $[n]$. We have:

$$\begin{aligned}
|M^{(n)}_{1,[n]}(1) - M^{(n)}_{2,[n]}(1)| &= M^{(n)}_{1,[n]}(1) - M^{(n)}_{2,[n]}(1) = \frac{n}{2n-1} - \frac{n}{(2+\varepsilon)n - (1+\varepsilon)} \\
&= \frac{n\left((2+\varepsilon)n - (1+\varepsilon)\right) - n(2n-1)}{(2n-1)((2+\varepsilon)n - (1+\varepsilon))} = \frac{\varepsilon n^2 - \varepsilon n}{(2n-1)((2+\varepsilon)n - (1+\varepsilon))} \\
&= \varepsilon \cdot \frac{n-1}{2n-1} \cdot \frac{n}{(2+\varepsilon)n - (1+\varepsilon)} \geq \varepsilon \cdot \frac{1}{3} \cdot \frac{n}{(2+\varepsilon)n} = \varepsilon \cdot \frac{1}{3} \cdot \frac{1}{(2+\varepsilon)} \\
&\geq \varepsilon \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{\varepsilon}{9},
\end{aligned}$$

where we used the fact that for any $x \geq 2$, $\frac{x-1}{2x-1} \geq \frac{1}{3}$. □

This entails that, if one were to use (Falahatgar et al., 2018, Theorem 12) to approximate the winning distribution of all the slates within ε, then, one would need to run their algorithm with $\varepsilon' \leq \frac{9\varepsilon}{n}$ incurring a cost of $\Omega(\frac{n^4 \log n}{\varepsilon^2})$ queries.

References

Tomer Adar. Tight simulation of a distribution using conditional samples. *arXiv*, 2506.18444, 2025.

Tomer Adar, Eldar Fischer, and Amit Levi. Optimal mass estimation in the conditional sampling model. In *SODA*, 2026.

Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: a survey. *JML*, 22(7):1–108, 2021.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

Clément L. Canonne. *A Survey on Distribution Testing: Your Data is Big. But is it Blue?* Number 9 in Graduate Surveys. Theory of Computing Library, 2020.

Clément L. Canonne, Dana Ron, and Rocco A. Servedio. Testing probability distributions using conditional samples. *SICOMP*, 44(3):540–616, 2015.

Clément L. Canonne. Topics and techniques in distribution testing: A biased but representative sample. *Foundations and Trends® in Communications and Information Theory*, 19(6):1032–1198, 2022.

Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. In *ITCS*, pages 561–580, 2013.

Arghya Roy Chaudhuri and Shivaram Kalyanakrishnan. PAC identification of many good arms in stochastic multi-armed bandits. In *ICML*, 2019.

Pinhan Chen, Chao Gao, and Anderson Y Zhang. Optimal full ranking from pairwise comparisons. *The Annals of Statistics*, 50(3):1775–1805, 2022.

Xi Chen, Sivakanth Gopi, Jieming Mao, and Jon Schneider. Competitive analysis of the top-k ranking problem. In *SODA*, pages 1245–1264, 2017.

Xi Chen, Yuanzhi Li, and Jieming Mao. A nearly instance optimal algorithm for top-k ranking under the multinomial logit model. In *SODA*, pages 2504–2522, 2018.

Yuxin Chen and Changho Suh. Spectral MLE: Top-K rank aggregation from pairwise comparisons. In *ICML*, pages 371–380, 2015.

Flavio Chierichetti, Mirko Giacchini, Ravi Kumar, Alessandro Panconesi, and Andrew Tomkins. Tight bounds for learning RUMs from small slates. In *NeurIPS*, pages 105864–105886, 2024.

Flavio Chierichetti, Mirko Giacchini, Ravi Kumar, Silvio Lattanzi, Alessandro Panconesi, Erasmo Tani, and Andrew Tomkins. Learning Multinomial Logits in O(n log n) Time. In *ICALP*, volume 374, 2026.

Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 1st edition, 2009.

Otto Dykstra. Rank analysis of incomplete block designs: A method of paired comparisons employing unequal repetitions on pairs. *Biometrics*, 16(2):176–188, 1960.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *COLT*, pages 255–270, 2002.

Moein Falahatgar, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Maximum selection and ranking under noisy comparisons. In *ICML*, pages 1088–1096, 2017.

Moein Falahatgar, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. The limits of maxing, ranking, and preference learning. In *ICML*, pages 1427–1436, 2018.

Lester R. Ford Jr. Solution of a ranking problem from binary comparisons. *The American Mathematical Monthly*, 64(8P2):28–33, 1957.

Ruijian Han, Rougang Ye, Chunxi Tan, and Kani Chen. Asymptotic theory of sparse Bradley–Terry model. *Annals of Applied Probability*, 30:2491–2515, 2020.

Minje Jang, Sunghyun Kim, Changho Suh, and Sewoong Oh. Optimal sample complexity of m-wise data for top-k ranking. In *NIPS*, volume 30, 2017.

Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *ICML*, pages 655–662, 2012.

Shimon Kogan and Merav Parter. New diameter-reducing shortcuts and directed hopsets: Breaking the $O(\sqrt{n})$ barrier. In *SODA*, 2022.

Wanshan Li, Shamindra Shrotriya, and Alessandro Rinaldo. ℓ_∞-bounds of the MLE in the BTL model under general comparison graphs. In *UAI*, 2022.

Anuran Makur and Japneet Singh. Minimax hypothesis testing for the bradley–terry–luce model. *IEEE Transactions on Information Theory*, 71(12):9163–9202, 2025.

Shie Mannor and John N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *JMLR*, 5:623–648, 2004.

Lucas Maystre and Matthias Grossglauser. Fast and accurate inference of Plackett–Luce models. In *NIPS*, volume 28, 2015.

Kuldeep S Meel, Gunjan Kumar, and Yash Pote. Distance estimation for high-dimensional discrete distributions. In *AISTATS*, pages 955–963, 2025.

Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pair-wise comparisons. In *NIPS*, 2012.

Sahand Negahban, Sewoong Oh, and Devavrat Shah. Rank centrality: Ranking from pairwise comparisons. *Oper. Res.*, 65(1), 2017.

Mark EJ Newman. Efficient computation of rankings from pairwise comparisons. *JMLR*, 24(238): 1–25, 2023.

Sam Olesker-Taylor and Luca Zanetti. An analysis of Elo rating systems via Markov chains. In *NeurIPS*, 2024.

Pinki Pradhan and Sampriti Roy. Distribution testing meets sum estimation. *arXiv*, 2504.15153, 2025.

Charvi Rastogi, Sivaraman Balakrishnan, Nihar B. Shah, and Aarti Singh. Two-sample testing on ranked preference data and the role of modeling assumptions. *J. Mach. Learn. Res.*, 23(1), 2022.

Wenbo Ren, Jia Liu, and Ness B. Shroff. On sample complexity upper and lower bounds for exact ranking from noisy comparisons. In *NIPS*, 2019.

Arjun Seshadri and Johan Ugander. Fundamental limits of testing the independence of irrelevant alternatives in discrete choice. In *EC*, page 65–66, 2019.

Arjun Seshadri, Alex Peysakhovich, and Johan Ugander. Discovering context effects from raw choice data. In *ICML*, pages 5660–5669, 2019.

Arjun Seshadri, Stephen Ragain, and Johan Ugander. Learning rich rankings. In *NIPS*, 2020.

Nihar Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramchandran, and Martin Wainwright. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. *JMLR*, 17(58):1–47, 2016.

Gordon Simons and Yi-Ching Yao. Asymptotics when the number of parameters tends to infinity in the Bradley–Terry model for paired comparisons. *The Annals of Statistics*, 27(3):1041–1060, 1999.

Balázs Szörényi, Róbert Busa-Fekete, Adil Paul, and Eyke Hüllermeier. Online rank elicitation for Plackett–Luce: A dueling bandits approach. In *NIPS*, 2015.

Kenneth E Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2003.

Ting Yan, Yaning Yang, and Jinfeng Xu. Sparse paired comparisons in the Bradley–Terry model. *Statistica Sinica*, 22:1305–1318, 09 2012.

Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *JCSS*, 78(5):1538–1556, 2012.

Ernst Zermelo. Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29(1):436–460, 1929.