Beyond the Full Slate: Evaluating MNL Algorithms on All Slates

Flavio Chierichetti[*]
Reddit
flavio.chierichetti@gmail.com

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

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

Silvio Lattanzi
Google, Barcelona
silviol@google.com

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

Erasmo Tani[*]
Cornell University
et484@cornell.edu

Andrew Tomkins
Google, Mountain View
atomkins@gmail.com

Abstract

Multinomial logit models (MNLs) are ubiquitous in machine learning, particularly in recommendation systems. While these models are typically evaluated using global metrics over a full catalog of items (such as ℓ_2-error), practical applications often involve making predictions on small, curated subsets or "slates" of items. A model that minimizes global error may nevertheless yield very poor predictions on specific sub-universes. Yet, evaluating performance across all possible subsets remains computationally challenging. In this paper, we present a set of new efficient algorithms to evaluate the small-slate performance of MNL models. Using these tools, we conduct an empirical study comparing standard non-adaptive and adaptive learning algorithms. We demonstrate that while methods that compute the Maximum Likelihood Estimate (MLE) suffer from poor worst-case performance on small slates, specialized adaptive algorithms that dynamically sample difficult subsets can mitigate this issue. Conversely, we find that non-adaptive algorithms often achieve competitive average-case performance; we provide a theoretical model explaining this phenomenon. We conclude with recommendations for practitioners regarding when to utilize adaptive sampling based on the specific robustness requirements of their application.

1 Introduction

Mathematical models for discrete choice capture the behavior of agents who must select one element from a discrete set of options (McFadden, 1974; Train, 2003). With the rise of the web economy, these models have taken a central role in many online applications, where they model interactions between users and platforms, and can be used to predict, for instance, the ads a user will click on, or the movie they will watch on a streaming platform.

Multinomial Logits (MNLs) are perhaps the most popular class of such models due to their simplicity and tractability (Bradley and Terry, 1952; Luce, 1959; Train, 2003). In an MNL M, we are given a

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

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

collection of n items, each item i equipped with a positive weight w_i.[2] When a user is presented with a *slate* $S \subseteq \{1, \ldots, n\}$ of options, M predicts that the user will select item $i \in S$ with probability $M_S(i)$ proportional to the item's weight:

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

This induces a conditional distribution M_S over the items in S called the *winning distribution*, representing the likelihood that each item will be selected as the winner among the options in the slate.

Evaluating MNL performance on all slates. In the platforms to which such models are applied, it is common to have an enormous catalog of items such as ads or movies. The catalog is too large to show to users in its entirety. Instead, platforms select smaller subsets of options that are manageable in size, allowing users to consider each element before deciding. This motivates the definition above of M_S for a specific slate S, rather than just for the catalog overall. The slates of interest to the platform vary widely depending on specifics of the business, and can be hard to predict in advance: the product team may choose to show carousels of action movies, dramas, as well as more esoteric groupings like movies similar to "Avatar," or movies whose main character is a cat. Ideally, the learned MNL M will perform well on all such slates, including the unknown ones that the product team may create after the learning algorithm has terminated. In evaluating a candidate MNL, it is natural to ask whether it will perform well on *any* slate to which it applies. To quantify this intuition, Chierichetti et al. (2026) define distance measures between MNLs M and M' as follows:

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

where $[n] = \{1, \ldots, n\}$. These measures capture the maximum prediction error incurred on any of the slates, either in ℓ_∞- or ℓ_1-norm.

While these measures seem natural, the state of the art in evaluation of MNLs does not generally consider worst-case slates. Instead, it is common to measure how well the learned MNL M' matches the true MNL M by considering $\|M_{[n]} - M'_{[n]}\|_p$ (typically with $p \in \{1, 2, \infty\}$). This is equivalent to comparing the predictions of the learned and true MNLs on the full catalog. In particular, it does not penalize smaller slates that might be important in practice, even if the predictions of the two MNLs on some of these slates are vastly different. For each $p \geq 1$, it is easy to construct such MNLs M, M', and M'' such that $\|M_{[n]} - M'_{[n]}\|_p \leq \|M_{[n]} - M''_{[n]}\|_p$ but $d_\infty(M, M') \gg d_\infty(M, M'')$ (see Appendix H).

We also note that downstream tasks sometimes require only the (approximate) top-k items of largest weight (Mohajer et al., 2017). In this setting, obtaining small d_∞ and d_1 suffices: if $d_1(M, M')$ is small, then the top-k items of M and M' must necessarily be approximately the same. Moreover, in general, a small worst-case error yields robustness across a wide range of downstream tasks, ensuring that improvements transfer simultaneously to multiple practical use cases.

Part of the reason that worst-case measures are not generally considered is that computing these measures appears to be very expensive: there are $\Theta(n^k)$ slates of size k, so considering each in turn may be prohibitive. Our primary contribution in this work is to circumvent this obstacle by developing a series of new algorithms to compute worst-case similarity measures of MNLs efficiently.

Given such algorithms, we then perform empirical comparisons to understand the performance of different MNL learning algorithms according to these worst-case measures. We first describe the algorithms we will compare:

Adaptive and non-adaptive MNL learning algorithms. The problem of learning MNLs has received significant attention (Negahban et al., 2012; Maystre and Grossglauser, 2015; Shah et al., 2016; Agarwal et al., 2018; Seshadri et al., 2019, 2020; Hendrickx et al., 2020; Olesker-Taylor and Zanetti, 2024). In the standard setting, an algorithm takes as input a training set consisting of a sequence of slates, each with the item selected by the user from that slate. The algorithm then outputs a vector $\hat{w} \in \mathbb{R}^n_{>0}$ of weights that represents an approximation of the true underlying MNL.[3]

[2] In practice, this weight might be a function of features of the item, the context, and the user; we consider the simplified setting of fixed non-parametric weights as previous works (Negahban et al., 2012; Hendrickx et al., 2020).

[3] Most MNL learning algorithms assume the subsets presented to the users are pairs, as this covers many important practical scenarios, and greatly simplifies computation.

Typically, $\hat{w}$ is learned through maximum likelihood estimation. We consider two state of the art algorithms for MNL learning: *Iterative Luce Spectral Ranking* (ILSR) (Maystre and Grossglauser, 2015) and *Minorization-Maximization* (MM) (Hunter, 2004), as these algorithms have been shown to outperform SGD for the MNL learning task.

Recently, Chierichetti et al. (2026) showed that any algorithm receiving uniform random queries will require $\Omega(n^2 \log n/\varepsilon^2)$ such samples in order to approximate the winning distribution for all pairs. To circumvent this lower bound, they design an algorithm *Lobster* that learns by selecting pairs to query adaptively and produces an approximation of the MNL to within d_1 distance ε in time $O((n/\varepsilon^3)\log n)$. In our empirical study, we therefore consider the adaptive algorithm *Lobster* as well as the non-adaptive ILSR and MM algorithms.

1.1 Contribution 1: Efficient Worst-case Measures

The measures $d_\infty(M, M')$ and $d_1(M, M')$ defined above are NP-hard to compute (see Appendix B). Our main algorithmic contributions in this paper are a set of approximation algorithms with guaranteed runtime, and a set of practical heuristics that work well on real datasets (Section 2). Specifically, we provide an algorithm that returns an approximation to the value $d_\infty(M, M')$ (resp., $d_1(M, M')$) to within an arbitrarily small additive error $\varepsilon \in (0, 1)$, in time $O(\frac{n^2}{\varepsilon^3})$ (resp., $n^{O(1/\varepsilon^2)}$). We also provide fast heuristics to find lower and upper bounds to the values of d_∞ and d_1 that scale to large datasets. For the case of pairs (slates of size 2), we consider the following metric:

$$d_\infty^{\text{pair}}(M, M') := \max_{\{i,j\}\subseteq[n]} \|M_{\{i,j\}} - M'_{\{i,j\}}\|_\infty,$$

which measures the distance on the worst-case pair of items. (An analogous definition of d_1^{pair} yields $d_1^{\text{pair}}(M, M') = 2 \cdot d_\infty^{\text{pair}}(M, M')$). We provide an algorithm that approximates $d_\infty^{\text{pair}}(M, M')$ within an additive error of ε in time $O((n/\varepsilon)\log n)$ improving upon the trivial running time of $O(n^2)$, which would not scale to the largest datasets in our analysis. We believe this collection of novel MNL quality measurement techniques will be essential for future research in MNL learning.

1.2 Contribution 2: Empirical Findings

With these measurement procedures in hand, we compare the different MNL learning algorithms by their performance on the full slate as well as their performance under the worst-case $d_\infty(M, M')$ and $d_1(M, M')$ measures, and the corresponding pair-only measures. We report four findings:

Finding 1: In our first set of experiments (Section 3.2), we focus on the worst-case error obtained over all slates. A first key finding is that, on datasets with a small number of items (n), non-adaptive algorithms obtain a smaller d_1-error (and d_∞-error) than adaptive ones. However, as n increases, the advantage of adaptivity emerges. The switch in behavior happens at $n \sim 10^3$—likely due to the hidden constants in the asymptotic guarantees of the adaptive algorithms. Our study thus reveals when adaptive algorithms achieve meaningful practical advantages, an insight that could not be obtained from theoretical guarantees alone. We match this observation with a theoretical finding showing that non-adaptive algorithms with access to $o(n^2)$ samples may see large errors on $\Omega(n)$ pairs (Appendix H). On the other hand, it is known (Han et al., 2020) that if all pairs of MNL weights have bounded ratio, then MLE will converge efficiently with good worst-case performance. In our experiments, we observe that smaller datasets exhibit smaller weight ratios, which further explains why non-adaptive approaches tend to outperform adaptive ones on those small datasets. We highlight that adaptive algorithms require active selection of the slates to query and are not implementable in settings where this is not allowed.

Finding 2: We observe that, even in real datasets, the error measured on the full slate $[n]$ is not representative of the worst-case error over all the slates, showing that indeed explicitly approximating the worst-case error is needed to understand the accuracy of future predictions. Interestingly, we find that the worst-case error on pairs is often a good indicator of the worst-case error on all slates. This is instrumental in practice, since the worst-case error on all pairs can be estimated more efficiently.

Finding 3: Given this observation, which suggests that slates of size two are informative in practice, we perform further experiments to study the behavior of the MNL learning algorithms on pairs (Appendix J.6). We find that non-adaptive algorithms can incur large errors even between pairs of items of similar weight, which are crucial in some practical applications. On the other hand, we find

that these difficult pairs are rare, and the algorithms perform remarkably well for the vast majority of the pairs.

Finding 4: We explain this latter behavior by analytically showing that $\tilde{O}(n)$ non-adaptive samples[4] suffice to approximate an MNL on all but a vanishing fraction of the pairs (Section 4), bypassing a known lower bound of $\Omega(n^2)$ that applies to the worst-case error (see Chierichetti et al. (2026)). This result, together with the empirical analysis, shows that non-adaptivity is sufficient for the average case, while adaptivity is preferable when approximating *all* the distributions is required.

All proofs missing from the main body and related work can be found in the Appendix.

2 Algorithms for Distance Approximation

In this section, we discuss algorithms that, given access to the weights $w_1, \ldots, w_n$ and $v_1, \ldots, v_n$ of MNLs M and M', approximate the quantities $d_\infty(M, M')$, $d_1(M, M')$, and d_1^{pair} (or d_∞^{pair}) to within an additive error $\varepsilon \in (0, 1)$. We begin by noting that computing $d_1(\cdot, \cdot)$ and $d_\infty(\cdot, \cdot)$ is NP-hard. In Appendix B, we show:

Theorem 2.1. *The problem of computing $d_1(M, M')$ (resp., $d_\infty(M, M')$) from the weight vectors w and v of M and M' is (weakly) NP-hard.*

This motivates our focus on obtaining approximation algorithms for these problems.

2.1 Algorithm for Approximating d_∞

We now describe an algorithm to approximate $d_\infty(M, M')$. Beyond being a contribution in its own right, this result serves as a conceptual stepping stone to obtain a practical algorithm to approximate d_∞^{pair}, which is essential for our empirical analysis. At a high level, our algorithm first fixes an item $i \in [n]$ and then finds a slate S that (approximately) maximizes $|M_S(i) - M'_S(i)|$. To find such a slate, it guesses an upper bound on the value of $v(S)$ and then maximizes the value of $w(S)$.[5] As we will show, this task is closely related to solving the following instance of the Knapsack problem, where $\alpha \in (0, 1]$:

$$\max_{S \subseteq [n] \setminus \{i\}} \quad \sum_{j \in S} w_j \quad \text{s.t.} \quad \sum_{j \in S} v_j \leq \left(\frac{1}{\alpha} - 1\right) \cdot v_i. \qquad (\texttt{Knapsack}(i, \alpha, M, M'))$$

Knapsack has a long history and there are now many fully-polynomial-time approximation schemes (FPTAS) for solving this problem. Let $\kappa(n, \varepsilon)$ be the optimal running time of an algorithm that outputs a $(1 - \varepsilon)$-approximation to $\texttt{Knapsack}(i, \alpha, M, M')$. A classic algorithm by Kellerer and Pferschy (1999) shows that $\kappa(n, \varepsilon) = O(n/\varepsilon^2)$.

The final algorithm consists of trying all possible values of $i \in [n]$, and $\alpha \in \{0, \Theta(\varepsilon), 2 \cdot \Theta(\varepsilon), \ldots, 1\}$. For each pair (i, α) it solves the corresponding instance of $\texttt{Knapsack}(i, \alpha, M, M')$ and returns the S that maximizes d_∞ among those tested. It can be shown that this algorithm provides the following guarantees (see Appendix C):

Theorem 2.2. *There is an algorithm that, given the weights of two MNLs M and M' supported on $[n]$ and an approximation parameter $\varepsilon \in (0, 1/3)$, in time $O\left(\frac{n}{\varepsilon} \cdot \kappa(n, \varepsilon/4)\right)$ outputs a slate $S \subseteq [n]$ and an item $i \in S$ such that:*

$$d_\infty(M, M') - \varepsilon \leq |M_S(i) - M'_S(i)| \leq d_\infty(M, M'). \qquad (1)$$

By using (Kellerer and Pferschy, 1999) to approximate Knapsack, this algorithm runs in time $O(\frac{n^2}{\varepsilon^3})$.

2.2 Fast Algorithm for Approximating d_∞^{pair}

We now turn to the problem of approximating $d_\infty^{\text{pair}}(M, M')$. Note that, since $d_1^{\text{pair}}(M, M') = 2 \cdot d_\infty^{\text{pair}}(M, M')$, this algorithm also approximates d_1^{pair}. Unlike the case of $d_\infty(\cdot, \cdot)$, a trivial

[4] Here, the notation $\tilde{O}(\cdot)$ is used to hide polylogarithmic factors.

[5] Given a vector $\{v_i\}_{i \in [n]}$ of numbers and a subset $S \subseteq [n]$ of indices, we define $v(S) := \sum_{i \in S} v_i$.

algorithm can find the exact value of $d_\infty^{\text{pair}}(M, M')$ in $O(n^2)$ time. However, the quadratic running time makes this algorithm too slow to be practical for large datasets.

We now obtain an efficient algorithm for this problem by using the idea described in Section 2.1. At a high level, if we restrict to pairs, $\texttt{Knapsack}(i, \alpha, M, M')$ becomes equivalent to $\max_{j \in [n] \setminus \{i\}} w_j$ s.t. $v_j \leq (1/\alpha - 1)v_i$. Observe that for a fixed i, if it holds that $v_j \leq v_k$ and $w_j \geq w_k$ then, j will necessarily be a better solution (than k) to $\texttt{Knapsack}(i, \alpha, M, M')$. Thus, our algorithm first calls a procedure that restricts the given instance to a set I such that for each $j, k \in I$ it holds that $j < k \iff w_j \leq w_k \iff v_j \leq v_k$, and then solves $\texttt{Knapsack}(i, \alpha, M, M')$ in time $O(\log n)$ by doing a binary search on the restricted instance I. In light of the previous observation, solving the problem on I will yield an optimal value to the original instance, since the filtering procedure only removes items that are strictly worse than some other item in I. Formally, we show the following (see Appendix D):

Theorem 2.3. *There is an algorithm that, given the weights of two MNLs M and M' supported on $[n]$ and an accuracy parameter $\varepsilon \in (0, 1/3)$, in time $O(\frac{n}{\varepsilon} \log n)$ returns a pair $\{i, j\}$ such that:*

$$d_\infty^{\text{pair}}(M, M') - \varepsilon \leq \|M_{\{i,j\}} - M'_{\{i,j\}}\|_\infty \leq d_\infty^{\text{pair}}(M, M').$$

2.3 Algorithm for Approximating d_1

We complete the picture by designing an algorithm that approximates d_1 to within an additive error of ε in polynomial time for constant ε. The algorithm leverages the fact that:

$$d_1(M, M') = 2 \cdot \max_{T \subseteq S \subseteq [n]} \left| \frac{w(T)}{w(S)} - \frac{v(T)}{v(S)} \right|.$$

The main idea is to guess the values of $w(T), w(S), v(T)$, and $v(S)$ and then find sets $T \subseteq S$ that approximately induce such values. To find such sets, we (i) eliminate items of large weight, (ii) construct a linear programming (LP) relaxation for the problem of fitting the remaining items, and (iii) stochastically round the LP solution to obtain sets $T \subseteq S$ that are approximately feasible. The complete procedure to find the approximate sets $T \subseteq S$ runs in expected $n^{O(1/\varepsilon^2)}$ time. We prove the following (see Appendix E):

Theorem 2.4. *There is an algorithm that, given weights $v_1, \ldots, v_n$ and $w_1, \ldots, w_n$ corresponding to MNLs M_1 and M_2, and an accuracy parameter $\varepsilon \in (0, 1/3)$, in* expected *time $n^{O(1/\varepsilon^2)}$ finds subsets $T \subseteq S \subseteq [n]$ such that:*

$$2 \cdot \left| \frac{w(T)}{w(S)} - \frac{v(T)}{v(S)} \right| \geq d_1(M_1, M_2) - \varepsilon.$$

2.4 An Efficient Upper Bound on d_1

While the algorithm in Section 2.3 guarantees an approximation of d_1 within an arbitrarily small additive error $\varepsilon > 0$, its running time is too high to be practical even for modest values of ε. Note that even without this algorithm, one can still find a slate S with a large value of $\|M_S - M'_S\|_1$, but upon finding such a slate S, it remains unclear how far its value is from the optimum.

Hence, we propose a fast and practical algorithm that yields a valid upper bound on $d_1(M, M')$. Our approach is inspired by the adaptive learning algorithm of (Chierichetti et al., 2026). This algorithm first (i) clusters items by weight and then (ii) builds an *estimation-forest* over the items, where the ratio of nearby items are accurately estimated and distant ones differ enough to not require precise estimation. This forest certifies the worst-case error between the MNL learned by the algorithm M' and the underlying true MNL M. Our idea to find a good upper bound to $d_1(M, M')$ is to construct a similar certificate, but in a setting where both M and M' can be arbitrary and do not explicitly provide an estimation-forest structure.

At a high level, our algorithm first guesses a partitioning $(R_1, \ldots, R_N)$ of the items $[n]$, mirroring the clustering of the adaptive algorithm. Then, for each $i \in [N]$, the algorithm upper bounds the worst-case error over slates containing items in R_i. It does so by splitting items in $R_i \cup \cdots \cup R_N$ into a head H and a tail T so that (i) replacing weights from M with those from M' (and vice versa)

induces only bounded multiplicative error on pairs in H, and (ii) items in T are negligible compared to those in R_i. This mirrors the distance separation of the items in the estimation-forest.

Our algorithm runs in time $O(n \log n)$ and outputs a valid upper bound, which we find to be informative in practice. A complete description of the algorithm is provided in Appendix F.

3 Experiments

In this section, we experimentally evaluate the approximation quality of different algorithms for learning MNLs.[6] In Section 3.2 we consider a (known) real-world MNL M and use a learning algorithm to obtain an estimate M', and then evaluate the distance between M and M'. As discussed, most prior work assessed the quality of M' by looking only at some distance error measure over the full slate. Theoretically, the error on the full slate can be significantly smaller than the error on the worst slate; however, this might not hold on real-world datasets. A central goal of this analysis—enabled by the algorithms developed in this paper—is to understand whether the error on the full slate is representative of the error on the worst-case slate in practice.

We observe that the full slate error can significantly underestimate the worst slate error even in real-world settings. On the other hand, we observe that the error on the worst pair is a good indicator of the worst possible error. In Appendix J.6 we conduct more experiments to further investigate the error distribution over pairs.

We then compare non-adaptive and adaptive learning algorithms to understand when learners can benefit from adaptivity.

3.1 Experimental Setup

Datasets. The ELO rating system is a scoring system that has become the de facto standard across different sports and competitive games (Stern, 2024). It assigns a score to each player that models their probability of winning. If player A has score R_A and player B has score R_B then the estimated probability that A wins against B is $\frac{1}{1+10^{(R_B-R_A)/400}}$. Observe that this is equivalent to representing each player with rating R as the item of an MNL with weight $10^{R/400}$. Hence, for our experiment, we obtain ground-truth MNLs from the ELO scores of players from three different domains:

(i) The dataset `tennis` contains the ELO scores of ≈ 500 currently active pro tennis players from `tennisabstract.com`. This dataset comes in four versions: `tennis-men`, `tennis-women`, `tennis-men-2025`, `tennis-women-2025`, depending on whether men or women athletes have been considered and whether the score is computed using all matches or only those from 2025.

(ii) The dataset `chess` contains the ELO scores of $\approx 10^4$ titled chess players on `chess.com`, a popular chess platform. This dataset has two versions: `chess-blitz` and `chess-bullet`, depending on whether the ELO score has been calculated from the players' blitz-timed or bullet-timed games.

(iii) Codeforces[7] is a competitive programming platform where users can take part in contests and are scored based on their performances. The score is computed similarly to the ELO rating, but generalized to competitions with more than two players. We consider datasets `codeforces-full`, `codeforces-active`, and `codeforces-competing` that contain a number of users ranging from ≈ 50k to ≈ 800k, and that respectively contain all the users, users who were online at least once the month before data collection, and users who took part in at least one contest in the preceding month.

Learning algorithms. We use both adaptive and non-adaptive algorithms that learn an MNL from pairwise comparisons. The non-adaptive algorithms take as input a dataset of comparisons and then iteratively approximate the maximum-likelihood-estimator (MLE). These algorithms converge to similar weight estimates, and they differ mostly in their running time. Therefore, we primarily use the state-of-the-art algorithm *Iterative Luce Spectral Ranking* (ILSR) (Maystre and Grossglauser, 2015). However, for Codeforces, we use the slower *Minorization-Maximization* (MM) algorithm (Hunter, 2004) since it is more memory efficient.[8]

[6] Code available at: `https://github.com/mirkogiacchini/evaluatingMNLs`

[7] `https://codeforces.com`

[8] One could instead have used the Accelerated Spectral Ranking (ASR) algorithm of Agarwal et al. (2018) to approximate the MLE. However, on our datasets, ASR and ILSR showed no significant difference in convergence speed. We therefore chose ILSR and MM, for which well-tested Python implementations are available in the Choix library.

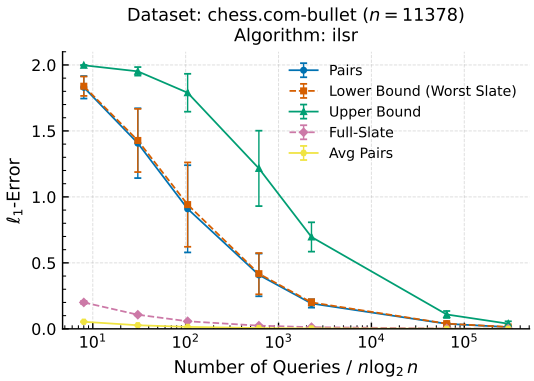


Figure 1: Distance between the MNL M of `chess-bullet` and an MNL M' learned by ILSR with varying budget of queries. We report: (i) d_1^{pair} (correct up to 0.01), (ii) a lower bound on d_1 computed via local search, (iii) an upper-bound on d_1, (iv) the full slate error $\|M_{[n]} - M'_{[n]}\|_1$, and (v) the average ℓ_1-error on pairs. The x-axis reports the number of queries normalized by $n \log_2 n$. Results are averaged over 10 runs and standard deviations are reported (randomness is over the sampling of the pairs on which M' is computed).

Our choice of adaptive algorithm is the *Lobster* algorithm proposed in (Chierichetti et al., 2026).[9] This is a natural choice, since it is provably asymptotically optimal for learning MNLs to minimize the worst-case error on all slates.

Evaluation algorithms. To compute d_1^{pair} we use our $O((n/\varepsilon)\log n)$ approximation algorithm (Theorem 2.3) with $\varepsilon = 0.01$. Before running this algorithm, we attempt to find the optimal solution directly by running a branch-and-bound heuristic for $O(n)$ steps. This heuristic has a pruning strategy that often allows it to run quickly in practice, despite its worst-case $\tilde{\Omega}(n^2)$ running time. In Appendix I, we provide more details on this heuristic and formal guarantees on its correctness. Specifically, if this procedure stops in $8n$ steps, we return the exact worst pair, otherwise we abort it and run the fast approximation algorithm instead. We find that the heuristic terminates quickly in many cases.

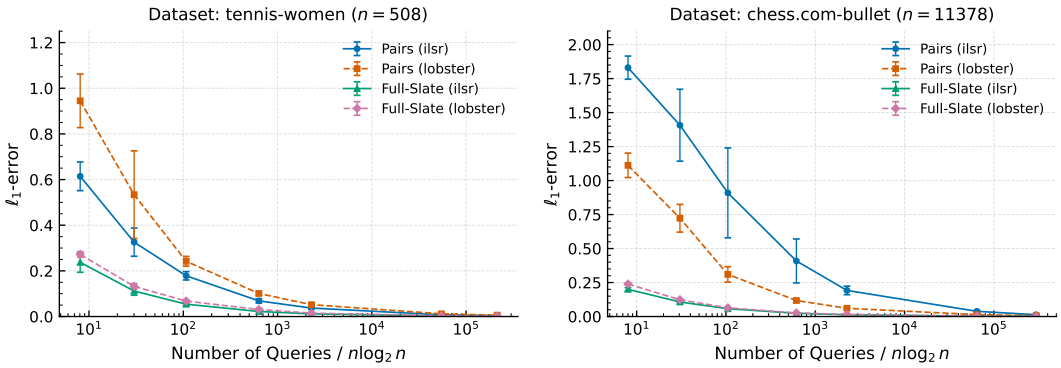
While polynomial, our approximation algorithms for d_∞ and d_1 are not fast enough to scale to thousands of items. Therefore, we use a local search algorithm to find a bad slate. This local search starts with computing the (approximately) worst pair, and then it greedily adds or removes one item if it increases the error. The local search is run for a maximum of $O(\log n)$ iterations to keep the running time feasible. The slate returned by the local search exhibits a lower bound on d_1 (or d_∞). To obtain an upper bound on d_1, we use the algorithm described in Section 2.4 (for an upper bound on d_∞, we note that $d_\infty(M, M') \le \frac{1}{2} d_1(M, M')$).

3.2 Error Between Real and Learned MNLs

In this experiment, we fix an MNL M, learn an MNL M' that is meant to approximate M, and evaluate the distance between M and M'. Specifically, the Lobster algorithm has an accuracy parameter ε that determines the quality of the learned MNL M'. Smaller values of ε require a larger number of (adaptive) queries. To ensure a fair comparison between adaptive and non-adaptive algorithms, for a fixed a value of ε, we first run Lobster and let L_ε be the number of queries it performs. Then, we run the non-adaptive algorithms on a dataset obtained by sampling L_ε pairs uniformly at random.

Note that if $L_\varepsilon = \Omega(n^2)$, then sampling L_ε pairs uniformly at random could require $\Theta(n^2)$ memory. This is prohibitive for the larger Codeforces datasets. Therefore, we use a different sampling scheme for this dataset: we first sample at most $Q \approx 10 \cdot n \ln n$ unique pairs, and then distribute the remaining query budget of $L_\varepsilon - Q$ uniformly among these pairs. This ensures that the dataset can be stored with $O(n \log n)$ memory. We use the following values of ε: $\{0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 1\}$.

[9] Each tree of the estimation-forest produced by this algorithm is a so-called lobster graph. We call this algorithm *Lobster* in our experiments for this reason.



(a) Results on `tennis-women` dataset. Here, the non-adaptive algorithm (ILSR) achieves smaller errors than the adaptive one (Lobster)

(b) Results on `chess-bullet` dataset. Here, the adaptive algorithm (Lobster) performs better than the non-adaptive one (ILSR)

Figure 2: Same setting as Figure 1, but we report the errors both for ILSR and Lobster. We report only the error on full slate and pairs.

Error on full slate vs. pairs. The results of this experiment on `chess-bullet` with ILSR are in Figure 1; the results for other datasets and algorithms are similar (Appendix J.3). We observe that the error on the full slate is much smaller than the error on the worst pair. Thus, empirically, the full slate is *not* representative of the quality of the learned MNL if approximating all the slates is important.

We also observe that the local search does not find a slate significantly worse than the worst pair. Moreover, the upper-bound on the error is always roughly $3\times$ larger than the lower bound. This suggests that the error on the worst-case pair is a *good* indicator of the quality of the learned MNL on all slates: when this value is small, in practice, the error on all slates tends to be small. For comparison, the ℓ_1-error (resp., ℓ_∞-error) on the full slate can be up to $50\times$ (resp., $1000\times$) smaller than our upper bound on d_1 (resp., d_∞). We observe also that the average error on pairs is instead very small even with few non-adaptive queries. In Appendix J.6 we perform further experiments to understand the error distribution over pairs and we observe that the vast majority of pairs incurs only a small error. We give a theoretical explanation of these findings in Theorem 4.3.

Adaptivity vs. non-adaptivity. We now turn to the question of understanding when adaptivity helps improving the approximation. We compare the performance of the Lobster algorithm and ILSR. We report the results for `tennis-women` and `chess-bullet` in Figure 2; the results for all other datasets are similar and can be found in Appendix J.3. We observe that for moderate domain sizes (`tennis-women`), the non-adaptive algorithm incurs smaller estimation errors than the adaptive one. However, as n increases, adaptivity wins: Lobster is significantly better than ILSR on `chess-bullet`. This behavior is likely due to the hidden constants of the adaptive algorithm: as n increases, the asymptotic superiority of adaptivity emerges.

To further validate this, we sparsify the datasets with at least 10^4 items and plot the difference in performance between ILSR and Lobster for varying number of items. We observe that Lobster starts performing better than ILSR at around 1000 items; see Appendix J.4 for details.

We observe that another reason for the superior performance of ILSR on tennis datasets is that the maximum ratio between two weights in the MNL is significantly smaller in tennis than in the other two datasets (see Appendix J.2).

Our evaluation algorithms are practically efficient: on the largest instance of Codeforces ($n \geq 800$k), they all run within a couple minutes, while on all other datasets they run within a few seconds.

4 Non-Adaptively Estimating Almost All Pairs

In this section, we prove that by non-adaptively querying an MNL on $\tilde{O}(n)$ pairs, one can accurately approximate the winning distribution on *almost all* pairs of items. (Interestingly, Chierichetti et al. (2026) showed that in order to approximate the winning distribution over *all* pairs, $\Omega(n^2 \log n/\varepsilon^2)$

non-adaptive queries are necessary.) This result yields a principled explanation for our otherwise surprising experimental findings (see Section 3.2).

Our algorithm leverages several properties of random graphs. Define $\binom{[n]}{2} := \{S \subseteq [n] \mid |S| = 2\}$. Let $G \sim \mathcal{G}(n,p)$ denote the Erdős–Rényi random graph with vertex set $[n]$, where each edge $\{i,j\} \in \binom{[n]}{2}$ is in G independently with probability p. Let $G \sim \mathsf{DAG}(n,p)$ denote the random directed acyclic graph (DAG) with vertex set $[n]$ obtained as follows: sample $G \sim \mathcal{G}(n,p)$ and orient each edge $\{i,j\}$ in G from i to j where $i < j$. In a DAG G, we say that a pair $\{i,j\} \in \binom{[n]}{2}$, with $i < j$, is *comparable* if there is a directed path from i to j in G. A classical result of Pittel and Tungol (2001) shows that most pairs are indeed comparable:

Lemma 4.1 ((Pittel and Tungol, 2001), Lemma 4.1 paraphrased)**.** *For any constant $A > 1$, let $G \sim \mathsf{DAG}(n,p)$, with $p = A \cdot \ln(n)/n$ and let F be the number of comparable pairs in G. Then,*

$$\mathbb{E}[F] = (1 \pm o(1))\frac{1}{2}\left(n\left(1 - \frac{1}{A}\right)\right)^2.$$

Our non-adaptive algorithm will sample a graph G from $\mathcal{G}(n,p)$, for an appropriate choice of p, and then query the MNL on each pair $\{i,j\} \in E(G)$ for $Q(n,\varepsilon)$ times in order to compute an estimate of the ratio w_i/w_j, for an appropriate choice of $Q(n,\varepsilon)$.

By renaming the items we can assume without loss of generality that $w_1 \leq \cdots \leq w_n$, where $\{w_i\}_{i\in[n]}$ are the weights of the hidden MNL. Therefore, for any comparable pair $\{i,j\}$, $i < j$, there exists a path $P = v_1, \ldots, v_\ell$ from i to j so that $w_{v_k} \leq w_{v_{k+1}}$ for any $k \in [\ell - 1]$. Thanks to this property, we will be able to compose the estimates made from the queries on each edge of the graph to estimate the probability $M_{\{i,j\}}(i)$. Indeed, we can have two cases:

Case 1: We have a good estimate for each ratio $w_{v_k}/w_{v_{k+1}}$ along the path. This implies that we can accurately estimate w_i/w_j, and therefore $M_{\{i,j\}}(i)$.

Case 2: There is a ratio $w_{v_k}/w_{v_{k+1}}$ along the path that is too costly to estimate accurately. But then we can conclude that $w_{v_k} \ll w_{v_{k+1}}$ and hence $w_i \ll w_j$, implying that approximating $M_{\{i,j\}}(i)$ by 0 is a good enough estimate.

Note that, in Case 1, the error in the estimate of the ratios compounds along the path. If the path P is too long, we might not end up with a good estimate for its endpoints. However, we can show that no path in a graph sampled from $\mathsf{DAG}(n,p)$ is too long. Specifically, we show:

Lemma 4.2. *Let $G \sim \mathsf{DAG}(n,p)$ with $p = A \cdot \ln(n)/n$ and $A \geq 1$. Then, the probability that there exists no directed path in G of length at least $e^2 \cdot A \cdot \ln n$ is at least $1 - O\left(\frac{1}{n}\right)$.*

In summary: if p is chosen sufficiently large in this process, most pairs of items will be connected with a directed path, and if p is sufficiently small, all directed paths in the graph will have to be short, allowing for accurate estimation, and constructing the graph will require few queries. Interestingly, we find a Goldilocks value of p that allows one to obtain all desired guarantees simultaneously.

By choosing $p \approx \frac{\ln^2 n}{n}$ and $Q(n,\varepsilon) \approx \frac{\ln^4 n}{\varepsilon^3}$, we show that, with high probability, the sampled graph has all the properties we need. We prove (see Appendix G):

Theorem 4.3. *Choose any $\varepsilon \in (0,1)$, and any constant $\delta \in (0,1)$, and let M be an MNL over $[n]$ for a large enough n. There exists a non-adaptive algorithm that, with probability $1 - o(1)$, makes at most $O\left(\frac{n \log^6 n}{\varepsilon^3}\right)$ queries to M on pairs and returns, for at least $(1 - \delta - o(1))\binom{n}{2}$ pairs $\{a,b\}$, an estimate $\hat{M}_{\{a,b\}}(a)$ of $M_{\{a,b\}}(a)$ satisfying $|\hat{M}_{\{a,b\}}(a) - M_{\{a,b\}}(a)| \leq \varepsilon$.*

Since the ℓ_∞-error on a single pair is at most 1, this result also immediately implies that the average error over all the pairs is at most $\varepsilon + \delta + o(1)$.

5 Conclusions, Limitations, and Future Work

In this work, we evaluated algorithms to learn MNLs, focusing on error metrics that capture estimation error across all slates, an aspect largely overlooked in prior work, yet crucial in many practical settings.

To enable this analysis, we developed theoretically sound algorithms and principled heuristics for approximating the worst-case error measures.

Through experiments on real-world datasets, we found that adaptive algorithms are generally superior when the goal is approximating all the slates. Instead, when the MNL involves only a small number of items, or when average-case error guarantees suffice, non-adaptive methods already perform well. This empirical observation is validated by a novel analysis showing that $O(n \log^6 n)$ non-adaptive queries suffice to accurately approximate almost all item pairs.

Limitations. Our experiments focus exclusively on competition datasets, rather than looking at more general application of MNLs. The reason for this is the following. Datasets for this task usually come in one of two forms: (i) either the explicit MNL weights are provided, or (ii) only pairs of (slate, user choice) are provided. We were particularly interested in using our evaluation algorithms to understand the difference in performance between adaptive and non-adaptive algorithms. The adaptive algorithm we consider needs to be able to query any possible slate, and this unfortunately rules out the possibility of using type-(ii) datasets. As a result, we focused on ELO-based datasets, since we are not aware of datasets in significantly different domains that are publicly available and provide explicit MNL weights. To partially mitigate this issue, we also performed experiments on a synthetic dataset mimicking product catalogs with heavy tailed weights. Results on this synthetic dataset are comparable to the results on our real-world datasets (see Appendix J.8 for details). Future work could improve upon this limitation. Other directions for future work include considering more expressive models such as Contextual MNLs.

Competing Interests. The authors report there are no competing interests to declare. The algorithms and empirical evaluation presented in this paper are research contributions and have not been deployed or integrated into any Reddit product or technology.

Acknowledgements

The authors thank the anonymous reviewers for the helpful comments.

Flavio Chierichetti was supported in part by BiCi – Bertinoro international Center for informatics, by a Google Focused Research Award and by the PRIN project 20229BCXNW (funded by the European Union - Next Generation EU, Mission 4 Component 1 CUP B53D23012910006).

Mirko Giacchini was supported in part by a fellowship of the Italian Academy for Advanced Studies in America at Columbia University.

Alessandro Panconesi was supported in part by BiCi – Bertinoro international Center for informatics.

References

Arpit Agarwal, Prathamesh Patil, and Shivani Agarwal. Accelerated spectral ranking. In *ICML*, 2018.

Michael Bender and Martin Farach-Colton. The LCA problem revisited. In *LATIN*, pages 88–94, 04 2000.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. A nearly quadratic-time FPTAS for knapsack. In *STOC*, 2024.

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

Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Light RUMs. In *ICML*, 2021.

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*, 2026.

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.

Bruce Hajek, Sewoong Oh, and Jiaming Xu. Minimax-optimal inference from partial rankings. In *NeurIPS*, 2014.

Ruijian Han, Rougang Ye, Chunxi Tan, and Kani Chen. Asymptotic theory of sparse Bradley–Terry model. *Ann. Appl. Prob.*, 30:2491–2515, 2020.

Julien Hendrickx, Alexander Olshevsky, and Venkatesh Saligrama. Graph resistance and learning from pairwise comparisons. In *ICML*, 2019.

Julien Hendrickx, Alex Olshevsky, and Venkatesh Saligrama. Minimax rate for learning from pairwise comparisons in the BTL model. In *ICML*, 2020.

David Hunter. MM algorithms for generalized Bradley–Terry models. *Ann. Stat.*, 32, 02 2004.

Hans Kellerer and Ulrich Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *J. Comb. Opt.*, 3:59–71, 1999.

Markus Leitner, Andrea Lodi, Roberto Roberti, and Claudio Sole. An exact method for (constrained) assortment optimization problems with product costs. *INFORMS J. Computs.*, 36(2):479–494, 2024.

Wanshan Li, Shamindra Shrotriya, and Alessandro Rinaldo. ℓ_∞-bounds of the MLE in the BTL model under general comparison graphs. In *UAI*, 2022.

R. Duncan Luce. *Individual Choice Behavior*. Wiley New York, 1959.

Xiao Mao. $(1-\varepsilon)$-approximation of knapsack in nearly quadratic time. In *STOC*, 2024.

Lucas Maystre and Matthias Grossglauser. Fast and accurate inference of Plackett–Luce models. In *NeurIPS*, 2015.

D. McFadden. Conditional logit analysis of qualitative choice behavior. Technical Report 105-142, Frontiers in Econometric, New York, 1974.

Soheil Mohajer, Changho Suh, and Adel Elmahdy. Active learning for top-k rank aggregation from noisy comparisons. In *ICML*, 2017.

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.

Sam Olesker-Taylor and Luca Zanetti. An analysis of Elo rating systems via Markov chains. In *NeurIPS*, 2024.

Boris Pittel and Ronald Tungol. A phase transition phenomenon in a random directed acyclic graph. *Rand. Struc. & Alg.*, 18(2):164–184, 2001.

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, 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 *NeurIPS*, 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. *Ann. Stat.*, 27(3):1041–1060, 1999.

Jacob Stern. A chess formula is taking over the world. *The Atlantic*, April 2024. URL https://www.theatlantic.com/technology/archive/2024/04/elo-ratings-are-everywhere/678129/. Accessed: 2026-01-17.

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.

A Related Work

MNLs have been studied for decades (Bradley and Terry, 1952; Luce, 1959; Train, 2003). The line of research most related to our work is that of MNL learning. Most of the works in this area focus on designing and analyzing algorithms that approximate the Maximum Likelihood Estimator. Their goal has been exclusively to obtain faster convergence rather than looking at novel ways to evaluate the quality of the final learned MNL (Maystre and Grossglauser, 2015; Negahban et al., 2017; Agarwal et al., 2018).

Indeed, most prior work evaluates the quality of the learned MNL by looking at some norm of the difference of the distributions over the full slate, both in theory and in practice (this is equivalent to looking at the norm of the difference of the normalized weight vectors). Specifically, Agarwal et al. (2018) evaluate the quality of the learned MNL by looking at $\|M_{[n]} - M'_{[n]}\|_1$, where M is the underlying MNL and M' is the learned MNL; Negahban et al. (2012, 2017); Hendrickx et al. (2019, 2020); Maystre and Grossglauser (2015) consider the norm $\|M_{[n]} - M'_{[n]}\|_2$ or a normalized version $\frac{\|M_{[n]} - M'_{[n]}\|_2}{\|M_{[n]}\|_2}$. Note that $M_{[n]}$ is equivalent to the vector of weights normalized to sum up to 1. However, as previously mentioned, these metrics can arbitrarily underestimate the prediction error incurred on some slates. As a simple example, consider the case where $w_1 := 1 - 2\varepsilon$, $w_2 = w_3 := \varepsilon$, and $w'_1 := 1 - 2\varepsilon$, $w'_2 := \frac{3}{2}\varepsilon$ and $w'_3 := \frac{1}{2}\varepsilon$ where $\varepsilon \in (0, 1)$ is arbitrarily close to 0. Then, $\|M_{[n]} - M'_{[n]}\|_1 = \varepsilon$, $\|M_{[n]} - M'_{[n]}\|_2 = \varepsilon/\sqrt{2}$, and $\frac{\|M_{[n]} - M'_{[n]}\|_2}{\|M_{[n]}\|_2} \approx \frac{\varepsilon}{\sqrt{2}(1-2\varepsilon)} = O(\varepsilon)$, while on the other hand, $\|M_{\{2,3\}} - M'_{\{2,3\}}\|_\infty = \frac{1}{4}$. The shortcomings on these metrics have also been observed in the recent papers of Han et al. (2020) and Chierichetti et al. (2026).

Another line of work considers the parametrization $\theta_i := \ln(w_i)$ such that $\sum_{i\in[n]} \theta_i = 0$, where $\{w_i\}_{i\in[n]}$ are the weights of an MNL M. Then, $M_S(i) = e^{\theta_i} / \sum_{j\in S} e^{\theta_j}$. Typically, previous works measure the quality of the MLE estimates by measuring $\|\theta - \theta'\|_2$, $\|\theta - \theta'\|_1$, $\|\theta - \theta'\|_\infty$, or normalized versions of these metrics (Li et al., 2022; Seshadri et al., 2020, 2019; Hajek et al., 2014). Note that these metrics can be unbounded even if the corresponding weights provide accurate estimates on all slates, therefore, they do not measure the prediction quality of the learned MNL and could be hard to interpret. As a simple example, consider $\theta_1 := 1000$, $\theta_2 := \varepsilon$, and $\theta'_1 = 10$, $\theta'_2 := \varepsilon$, then, these two sets of parameters provide roughly the same predictions, but after normalizing them so that they sum up to 0, they are at extremely large distance even in ℓ_∞-norm (and this distance can be increased arbitrarily by increasing the constant 1000). In this work, we mostly compare to metrics on the set of weight (not their logarithm), which are bounded as d_1 and d_∞ and more interpretable.

To the best of our knowledge, the paper of Chierichetti et al. (2026) is the only work that provides an adaptive learning algorithm instead of studying the quality of non-adaptive MLE and that explicitly optimizes for the worst prediction metrics on all slates. However, they do not consider the problem of computing these metrics, which is a non-trivial task by itself, and they do not evaluate their algorithms experimentally.

Our work differs from all of the previous ones in that we focus on the problem of evaluation of MNLs rather than learning. We evaluate MNLs under the metrics d_1 and d_∞, which require the learned MNL to provide accurate predictions on all slates. Using these new tools, we evaluate experimentally the quality of previously proposed learning algorithms.

We also mention that some works explicitly consider the goal of minimizing $|\frac{w_i}{w'_i} - 1|$ for each $i \in [n]$ (Han et al., 2020; Yan et al., 2012; Simons and Yao, 1999) and study the performance of MLE under this distance. These results are similar in spirit to our result in Section 4, as they also use tools from $G(n, p)$ random graphs. However, their theorems require a bound on the ratio of the maximum and minimum weights in the MNL, while our result applies to *any* MNL, without any assumption on its weights.

Several works also considered the problem of ranking the items of an MNL from pairwise queries (Ren et al., 2019; Chen et al., 2022; Falahatgar et al., 2017, 2018) or finding the top-k items of largest weights (Mohajer et al., 2017). Both tasks are different from learning since the ranking of the items alone does not allow to predict the degree to which an item is likely to win. Moreover, we observe that learning by minimizing worst case errors such as $d_1(M, M')$ produces weight estimates that can be used for (approximate) ranking and top-k as well.

We also mention that other works on MNLs have observed connections with Knapsack, for example in the context of assortment optimization (Leitner et al., 2024). However, to the best of our knowledge, we are the first to show approximation algorithms to compute the worst case metrics d_∞ and d_1 and connections with Knapsack in this area were not made before.

B Hardness of Computing d_1 and d_∞

In this section, we show that it is NP-hard to compute $d_1(M_1, M_2)$ and $d_\infty(M_1, M_2)$, from the weights of the two MNLs M_1 and M_2.

We will show NP-hardness of these problems via reductions from SUBSET-SUM. For completeness we recall the definition of this problem, which has long been known to be (Weakly) NP-hard.

SUBSET-SUM: Given a collection of positive integers $A = \{a_1, \ldots, a_n\}$ and a target positive integer K decide whether there exists a subset $S \subseteq [n]$ such that:

$$\sum_{i \in S} a_i = K.$$

We note that the problem remains NP-hard when $K \geq 2$ and we shall make the assumption that every instance satisfies this property.

Recall that, given a collection of numbers $\{v_i\}_{i\in[n]}$, and a subset $S \subseteq [n]$, we will write $v(S)$ for the value $v(S) := \sum_{i\in S} v_i$.

We now prove that exactly computing d_1 is NP-hard. This proof was developed with the assistance of Gemini, and later written and verified by the authors.

Theorem B.1. *The problem of exactly computing $d_1(M_1, M_2)$ from the weights of two MNLs M_1 and M_2 is (Weakly) NP-hard.*

Proof. We note that the optimization problem of computing d_1 is at least as hard as the following decision problem. We will therefore prove NP-hardness of this problem instead.

MAX-RATIO-DIFFERENCE (MRD): Given two ordered collections of n positive real numbers $\{v_i\}_{i\in[n]}$, and $\{w_i\}_{i\in[n]}$, and a target objective $\alpha \in \mathbb{R}_{\geq 0}$ decide whether there exists two subsets S and T of $[n]$ with $\varnothing \subset T \subset S \subseteq [n]$, such that:

$$R_{v,w}(S, T) := \frac{v(T)}{v(S)} - \frac{w(T)}{w(S)} \geq \alpha.$$

Indeed, $\exists\, \varnothing \subset T \subset S \subseteq [n]$ s.t. $R_{v,w}(S, T) \geq \alpha \iff d_1(M_1, M_2) \geq 2\alpha$, where $\{v_i\}_{i\in[n]}$ and $\{w_i\}_{i\in[n]}$ are the weights of the two MNLs and $\alpha \in [0, 1]$.

We construct a Karp reduction (a polynomial many-to-one reduction) from SUBSET-SUM to MRD. Given an instance $(A = \{a_1, \ldots, a_n\}, K)$ of SUBSET-SUM, we define $L = K^2$ and note that $L > 1$. We then define an instance $(\{v_i\}_{i\in[n+1]}, \{w_i\}_{i\in[n+1]}, \alpha)$ of MRD as follows:

$$\forall i \in [n] \quad v_i := a_i, \text{ and } w_i := a_i,$$

$$v_{n+1} = 1 \text{ and } w_{n+1} = L,$$

and $\alpha = \frac{K-1}{K+1}$. We note that the size of this MRD instance is polynomial in the size of the original SUBSET-SUM instance. We are therefore left with showing that this reduction preserves *yes* and *no* instances.

By construction, for any pair of sets (T, S) with $T \subseteq S$, if $n + 1 \notin S$, then $v(T) = w(T)$ and $v(S) = w(S)$ and hence:

$$\frac{v(T)}{v(S)} - \frac{w(T)}{w(S)} = 0 < \alpha.$$

Similarly, if $S = \{n + 1\}$ then it must be $T = S$ and thus $R_{v,w}(S, T) = 0 < \alpha$.

So the MRD instance is a *yes* instance if and only if there exists some sets (T, S) with $n+1 \in S$ and $S \cap [n] \neq \varnothing$ such that $R_{v,w}(S,T) \geq \alpha$.

Consider a set S with $n+1 \in S$. Let $S_{[n]} := S \cap [n]$. We have:

$$v(S) = v(S_{[n]}) + v_{n+1} = a(S_{[n]}) + 1,$$

while:

$$w(S) = w(S_{[n]}) + w_{n+1} = a(S_{[n]}) + L.$$

We are interested in finding a set T^* such that:

$$T^* \in \operatorname*{argmax}_{T \subseteq S} \frac{v(T)}{v(S)} - \frac{w(T)}{w(S)} = \operatorname*{argmax}_{T \subseteq S} \underbrace{(a(S_{[n]}) + L)v(T) - (a(S_{[n]}) + 1)w(T)}_{=:G(T)}.$$

Note that any such T^* must contain all elements of $S_{[n]}$, since they each contribute a positive amount to $G(T)$ (in fact, for each element $i \in [n]$ included in T, the value of the function $G(T)$ increases by $(L-1)a_i$). Moreover, the maximizing T^* cannot contain the element $n+1$, which would decrease the value of $G(T)$ by $|a(S_{[n]})(1-L)|$. Hence T^* must be equal to $S_{[n]}$, or, in other words:

$$\operatorname*{argmax}_{T \subseteq S} \frac{v(T)}{v(S)} - \frac{w(T)}{w(S)} = \left\{S_{[n]}\right\}.$$

So for each choice of S, the maximizing pair (T, S) is given by $(S_{[n]}, S)$ and has value:

$$\frac{v(S_{[n]})}{v(S)} - \frac{w(S_{[n]})}{w(S)} = \frac{a(S_{[n]})(L-1)}{(a(S_{[n]})+1)(a(S_{[n]})+L)}.$$

A straight-forward calculus argument shows that the function:

$$h(x) = \frac{x(L-1)}{(x+1)(x+L)}$$

over the domain $\mathbb{R}_{\geq 0}$, has a maximum value of α achieved at the unique maximizer: $x = \sqrt{L} = K$.

Hence, there exists two sets (T, S) with:

$$\frac{v(T)}{v(S)} - \frac{w(T)}{w(S)} \geq \alpha$$

if and only if there exists a subset $X \subseteq [n]$ (with $X = S_{[n]}$) such that:

$$a(X) = K,$$

showing that the reduction preserves *yes* and *no* instances as needed. □

We now provide a similar reduction for the case of d_∞. This proof was developed with the assistance of Gemini, and later written and verified by the authors.

Theorem B.2. *The problem of exactly computing $d_\infty(M_1, M_2)$ from the weights of two MNLs M_1 and M_2 is (Weakly) NP-hard.*

Proof. The problem of computing d_∞ is at least as hard as the following decision version of the problem, we shall therefore prove NP-hardness of this problem instead.

MAX-ABS-RATIO-DIFFERENCE (MARD): Given two ordered collections of n positive rational numbers $\{v_i\}_{i\in[n]}$ and $\{w_i\}_{i\in[n]}$, and a target objective $\alpha \in \mathbb{Q}_{>0}$, decide whether there exists a pair (S, i) where $S \subseteq [n]$ is a non-empty subset of $[n]$ and i is an element of S, such that:

$$R_{v,w}(S,i) := \left| \frac{v_i}{v(S)} - \frac{w_i}{w(S)} \right| \geq \alpha.$$

This proof follows a similar structure to that of Theorem B.1. We reduce from SUBSET-SUM. Given an instance $(A = \{a_i\}_{i\in[n]}, K)$ of SUBSET-SUM, the reduction constructs an instance of MARD as follows. We let $\{v_i\}_{i\in[n+1]}$ and $\{w_i\}_{i\in[n+1]}$ be given by:

$$\forall i \in [n]: \quad v_i = w_i = a_i$$

while $v_{n+1} = 2K$ and $w_{n+1} = K/2$, and $\alpha = \frac{1}{3}$. Note that this reduction is polynomial-time.

In this construction, there is always a maximizer of $R_{v,w}(S, i)$ where S is such that $n + 1 \in S$ and $S_{[n]} := S \cap [n] \neq \varnothing$. This is because if $S \subseteq [n]$ or $S = \{n + 1\}$ the values of $R_{v,w}(S, i)$ is zero. Fixing any such S, we have:

$$\begin{aligned}
\max_{i\in S_{[n]}} R_{v,w}(S,i) &= \max_{i\in S_{[n]}} \left|\frac{v_i}{v(S)} - \frac{w_i}{w(S)}\right| \\
&= \left|\frac{1}{v(S)} - \frac{1}{w(S)}\right| \max_{i\in S_{[n]}} a_i \\
&= \frac{\frac{3}{2}K}{v(S)w(S)} \max_{i\in S_{[n]}} a_i
\end{aligned}$$

while:

$$\begin{aligned}
R_{v,w}(S, n+1) &= \left|\frac{v_{n+1}}{v(S)} - \frac{w_{n+1}}{w(S)}\right| \\
&= \left|\frac{2K}{v(S_{[n]}) + 2K} - \frac{K/2}{w(S_{[n]}) + K/2}\right| \\
&= \frac{\frac{3}{2}K}{v(S)w(S)} a(S_{[n]}) \\
&\geq \max_{i\in S_{[n]}} R_{v,w}(S,i)
\end{aligned}$$

and hence:

$$n + 1 \in \operatorname*{argmax}_{i\in S} R_{v,w}(S,i)$$

and:

$$\begin{aligned}
\max_S \max_{i\in S} R_{v,w}(S,i) &= \max_S R_{v,w}(S, n+1) \\
&= \max_S \frac{\frac{3}{2}K}{v(S)w(S)} a(S_{[n]}) \\
&= \max_S \frac{\frac{3}{2}K}{(a(S_{[n]}) + 2K)(a(S_{[n]}) + K/2)} a(S_{[n]}).
\end{aligned}$$

Considering the function:

$$h(x) = \frac{\frac{3}{2}K}{(x + 2K)(x + K/2)} \cdot x$$

and applying a simple calculus argument, we find that over the domain $\mathbb{R}_{\geq 0}$ it achieves a unique maximum at:

$$x = K$$

and its maximum value is:

$$\frac{1}{3} = \alpha.$$

Hence, the MARD instance has a pair (S, i) with $R_{v,w}(S, i) \geq \alpha$ if and only if there exists some subset $S_{[n]} \subseteq [n]$ such that $a(S_{[n]}) = K$, and the reduction given is a valid Karp reduction. □

Algorithm 1 Approximate-$d_\infty(M, M', \varepsilon)$

Input: The weights $\{w_j\}_{j\in[n]}$ of an MNL M, the weights $\{v_j\}_{j\in[n]}$ of an MNL M', and an approximation parameter $\varepsilon \in (0, 1/3)$.
Output: A slate $S \subseteq [n]$ and an item $i \in S$ that give the approximate value of $d_\infty(M, M')$ (as in (1)).
$\varepsilon_1 = \varepsilon/2$
$\varepsilon_2 = \varepsilon/4$
Best$= 0$
$S_{\text{final}} = \varnothing$
$i_{\text{final}} = 0$
for $i \in [n]$ **do**
  for $\alpha \in \{0, \varepsilon_1, 2\varepsilon_1, \ldots, 1\}$ **do**
    if $\alpha = 0$ **then**
      $S = [n] \setminus \{i\}$
    else
      Compute a $(1 - \varepsilon_2)$-approximate solution S to $\texttt{Knapsack}(i, \alpha, M, M')$ (if feasible)
    end if
    if $M'_{S\cup\{i\}}(i) - M_{S\cup\{i\}}(i) >$Best **then**
      $S_{\text{final}} = S$
      $i_{\text{final}} = i$
      Best$= M'_{S\cup\{i\}}(i) - M_{S\cup\{i\}}(i)$
    end if
  end for
end for
Swap M and M' and repeat the **for** loop above.
return $S_{\text{final}} \cup \{i_{\text{final}}\}, i_{\text{final}}$

C Omitted Proofs from Section 2.1

In this section, we show that Algorithm 1 provides an estimate to d_∞ within an additive $\varepsilon \in (0, 1)$, in time $O(\frac{n}{\varepsilon} \cdot \kappa(n, \varepsilon/4))$. While using the algorithm of Kellerer and Pferschy (1999) results into a $O(n^2/\varepsilon^3)$ complexity, note that any FPTAS supporting Knapsack instances with non-integer weights and profits can be used in practice. In particular, if the weights of M and M' are integers (or rationals with polynomially many bits) then, the algorithms of (Mao, 2024; Chen et al., 2024) give an FPTAS for Knapsack with complexity $\tilde{O}(n + 1/\varepsilon^2)$ giving a running time of $\tilde{O}(\frac{n^2}{\varepsilon} + \frac{n}{\varepsilon^3})$ for approximating d_∞.

Theorem 2.2. *There is an algorithm that, given the weights of two MNLs M and M' supported on $[n]$ and an approximation parameter $\varepsilon \in (0, 1/3)$, in time $O\left(\frac{n}{\varepsilon} \cdot \kappa(n, \varepsilon/4)\right)$ outputs a slate $S \subseteq [n]$ and an item $i \in S$ such that:*

$$d_\infty(M, M') - \varepsilon \le |M_S(i) - M'_S(i)| \le d_\infty(M, M'). \tag{1}$$

Proof. We have:

$$\begin{aligned} d_\infty(M, M') &= \max_{\substack{S\subseteq[n]\\ S\neq\varnothing}} \|M_S - M'_S\|_\infty \\ &= \max_{\substack{S\subseteq[n]\\ S\neq\varnothing}} \max_{i\in S} |M_S(i) - M'_S(i)| \\ &= \max_{i\in[n]} \max_{S\subseteq[n]\setminus\{i\}} |M_{S\cup\{i\}}(i) - M'_{S\cup\{i\}}(i)|. \end{aligned}$$

The problem then reduces to that of approximating

$$\max_{S\subseteq[n]\setminus\{i\}} |M_{S\cup\{i\}}(i) - M'_{S\cup\{i\}}(i)|$$

for every $i \in [n]$.

Fixed $i \in [n]$, we consider the following optimization problem:

$$\begin{aligned} \min_{S \subseteq [n] \setminus \{i\}} \quad & M_{S \cup \{i\}}(i) \\ \text{s.t.} \quad & M'_{S \cup \{i\}}(i) \geq \alpha. \end{aligned} \qquad (\texttt{MinProb}(i, \alpha, M, M'))$$

Note that for $\alpha = 0$, the optimal solution is $S = [n] \setminus \{i\}$. Thus, let us consider the case $\alpha > 0$ in the remaining of the proof. Note that this problem is equivalent to $\texttt{Knapsack}(i, \alpha, M, M')$ in the following sense:

Claim C.1. *Every feasible solution S to $\texttt{Knapsack}(i, \alpha, M, M')$ is also a feasible solution to $\texttt{MinProb}(i, \alpha, M, M')$, and vice versa. Moreover, if S is a $(1-\varepsilon)$ multiplicative approximation to the optimum of $\texttt{Knapsack}(i, \alpha, M, M')$ for some $\varepsilon \leq 1/2$, then it is a $(1+2\varepsilon)$ multiplicative approximation to the optimum of $\texttt{MinProb}(i, \alpha, M, M')$.*

Proof of Claim C.1. Since $M'_{S \cup \{i\}}(i) = \frac{1}{1+\sum_{j \in S} v_j / v_i}$, we have $M'_{S \cup \{i\}}(i) \geq \alpha$ if and only if $\sum_{j \in S} v_j \leq \left(\frac{1}{\alpha} - 1\right) v_i$. Moreover, if $\sum_{j \in S} w_j \geq (1-\varepsilon) \sum_{j \in S'} w_j$ for some $\varepsilon \in [0, 1/2]$, then:

$$\begin{aligned} M_{S \cup \{i\}}(i) &= \frac{1}{1 + \sum_{j \in S} w_j / w_i} \\ &\leq \frac{1}{1 + (1-\varepsilon) \sum_{j \in S'} w_j / w_i} \\ &\leq \frac{1}{(1-\varepsilon)} \cdot \frac{1}{1 + \sum_{j \in S'} w_j / w_i} \\ &\leq (1+2\varepsilon) \cdot \frac{1}{1 + \sum_{j \in S'} w_j / w_i} \\ &= (1+2\varepsilon) \cdot M_{S' \cup \{i\}}(i) \end{aligned}$$

so that any $(1-\varepsilon)$-approximation to the optimum of $\texttt{Knapsack}(i, \alpha, M, M')$ is a $(1+2\varepsilon)$-approximation to the optimum of $\texttt{MinProb}(i, \alpha, M, M')$, concluding the proof of Claim C.1. □

We now show that Algorithm 1 satisfies the desired guarantees. Note that one of two things has to be true:

Case 1: We have:

$$d_\infty(M, M') = M'_{S^* \cup \{i^*\}}(i^*) - M_{S^* \cup \{i^*\}}(i^*)$$

for some S^* and i^*.

Case 2: Alternatively:

$$d_\infty(M, M') = M_{S^* \cup \{i^*\}}(i^*) - M'_{S^* \cup \{i^*\}}(i^*)$$

for some S^* and i^*.

We will now show that in Case 1, the first run through our algorithm correctly finds an S_{final} and i_{final} satisfying the conditions of Equation (1). By an analogous argument, our algorithm will satisfy the same guarantees at the end of the second run through the for loop (Line 22) in Case 2, and this will imply the statement of the theorem.

We make use of the following claim which we prove later:

Claim C.2. *Assuming Case 1, let S^* and i^* be the set and item such that:*

$$d_\infty(M, M') = M'_{S^* \cup \{i^*\}}(i^*) - M_{S^* \cup \{i^*\}}(i^*).$$

There exists some $\alpha \in \{0, \varepsilon_1, 2\varepsilon_1, \ldots, 1\}$ such that $\texttt{MinProb}(i^*, \alpha, M, M')$ admits a feasible solution and:

$$\alpha - \texttt{MinProb}(i^*, \alpha, M, M') \geq d_\infty(M, M') - \varepsilon_1.$$

Assume Case 1. At each iteration, our algorithm considers a specific value of i and α, finds a set S, and checks whether the value of

$$M'_{S\cup\{i\}}(i) - M_{S\cup\{i\}}(i)$$

is greater than that of all the sets found so far. It is therefore sufficient to show that there is an iteration in which the algorithm finds an S such that:

$$M'_{S\cup\{i\}}(i) - M_{S\cup\{i\}}(i) \geq d_\infty(M, M') - \varepsilon.$$

Consider i^* as in the statement of Claim C.2 and let α be the value guaranteed to exist by the claim. Let $S^{(\alpha,i^*)}$ be the approximate solution to $\texttt{Knapsack}(i, \alpha, M, M')$ found by the algorithm in the iteration corresponding to i^* and α.

$$\begin{aligned}
&M'_{S^{(\alpha,i^*)}\cup\{i^*\}}(i^*) - M_{S^{(\alpha,i^*)}\cup\{i^*\}}(i^*) \\
&\geq \alpha - (1+2\varepsilon_2)\texttt{MinProb}(i^*, \alpha, M, M') \\
&\geq \alpha - \texttt{MinProb}(i^*, \alpha, M, M') - 2\varepsilon_2 \\
&\geq d_\infty(M, M') - \varepsilon_1 - 2\varepsilon_2 \\
&\geq d_\infty(M, M') - \varepsilon.
\end{aligned}$$

Where the first inequality follows from Claim C.1 and by the $(1-\varepsilon_2)$-approximation guarantee, the second one follows from:

$$\texttt{MinProb}(i^*, \alpha, M, M') \leq 1$$

and the third follows from Claim C.2. This implies Theorem 2.2. We conclude by proving Claim C.2.

Proof of Claim C.2. Let $p = M'_{S^*\cup\{i^*\}}(i^*)$ and consider the unique $\ell \in \mathbb{N}\cup\{0\}$ such that $\ell\varepsilon_1 \leq p < (\ell+1)\varepsilon_1$ and consider $\alpha = \ell\varepsilon_1$. Since $M'_{S^*\cup\{i^*\}}(i^*) = p \geq \alpha$, S^* is a feasible solution to $\texttt{MinProb}(i^*, \alpha, M, M')$. Moreover, the optimal solution to the program is some set $T \subseteq [n]\setminus\{i^*\}$ such that:

$$M_{T\cup\{i^*\}}(i^*) \leq M_{S^*\cup\{i^*\}}(i^*)$$

this gives:

$$\begin{aligned}
&\alpha - \texttt{MinProb}(i^*, \alpha, M, M') \\
&= \alpha - M_{T\cup\{i^*\}}(i^*) \\
&\geq \alpha - M_{S^*\cup\{i^*\}}(i^*) \\
&\geq p - \varepsilon_1 - M_{S^*\cup\{i^*\}}(i^*) \\
&= M'_{S^*\cup\{i^*\}}(i^*) - M_{S^*\cup\{i^*\}}(i^*) - \varepsilon_1 \\
&= d_\infty(M, M') - \varepsilon_1,
\end{aligned}$$

concluding the proof of Claim C.2. □

D Omitted Proofs from Section 2.2

In this section we prove Theorem 2.3. Specifically, we prove that Algorithm 2, returns an ε-additive approximation of d_∞^{pair} in time $O(\frac{n\log n}{\varepsilon})$.

We begin by observing that the worst error on pairs can be written in a more convenient way.

Lemma D.1. *Let* $\text{val}^i(j) = M'_{\{i,j\}}(i) - M_{\{i,j\}}(i)$. *Then,*

$$d_\infty^{\text{pair}}(M, M') = \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} \text{val}^i(j). \tag{2}$$

Proof. We have:

$$\begin{aligned}
d_\infty^{\text{pair}}(M, M') &:= \max_{\{i,j\}\subseteq[n]} \|M_{\{i,j\}} - M'_{\{i,j\}}\|_\infty \\
&= \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} |M_{\{i,j\}}(i) - M'_{\{i,j\}}(i)|.
\end{aligned}$$

Algorithm 2 ApproximatePairDist(M, M', ε)

Input: The weights $\{w_i\}_{i\in[n]}$ of an MNL M, the weights $\{v_i\}_{i\in[n]}$ of an MNL M', and an approximation parameter $\varepsilon \in (0, 1/3)$.
We assume that $v_1 \le v_2 \le \cdots \le v_n$, otherwise we relabel the items accordingly. This can be done without loss of generality since this operation can be performed in $O(n \log n)$ time.
Output: A pair $\{i, j\}$ that is close to witnessing $d_\infty^{\text{pair}}(M, M')$.
$I = \texttt{Filter}((w_i)_{i\in[n]}, (v_i)_{i\in[n]})$
$\varepsilon_1 = \left\lceil \frac{1}{\varepsilon} \right\rceil^{-1}$
$best = 0$, $i_{\text{final}} = 1$, $j_{\text{final}} = 2$
for $i \in [n]$ **do**
  for $\alpha \in \{0, \varepsilon_1, 2\varepsilon_1, \ldots, 1\}$ **do**
    Using binary search, find the largest $j_{\alpha,i} \in I \setminus \{i\}$ such that $v_{j_{\alpha,i}} \le \frac{1-\alpha}{\alpha} v_i$ (if it exists)
    $\text{val} = M'_{i,j_{\alpha,i}}(i) - M_{i,j_{\alpha,i}}(i)$
    if $\text{val} > best$ **then**
      $best = \text{val}$
      $i_{\text{final}} = i$
      $j_{\text{final}} = j_{\alpha,i}$
    end if
  end for
end for
return $\{i_{\text{final}}, j_{\text{final}}\}$

Algorithm 3 $\texttt{Filter}((w_i)_{i\in[n]}, (v_i)_{i\in[n]})$

Input: The weights $(w_i)_{i\in[n]}$ of an MNL M, the weights $(v_i)_{i\in[n]}$ of an MNL M'. We assume $v_1 \le v_2 \le \cdots \le v_n$.
Output: A subset $I \subseteq [n]$ with the property that, for all $i, j \in I$ with $i \le j$, $w_i \le w_j$ and $v_i \le v_j$.
$i = 1, j = 2, I = \{1\}$
while $j \le n$ **do**
  if $w_i \ge w_j$ **then**
    $j = j + 1$
  else
    $I = I \cup \{j\}$
    $i = j$
    $j = i + 1$
  end if
end while
return I

Therefore, we have:

$$\begin{aligned} d_\infty^{\text{pair}}(M, M') &\ge \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} M'_{\{i,j\}}(i) - M_{\{i,j\}}(i) \\ &= \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} \text{val}^i(j), \end{aligned}$$

showing that the left-hand side of (2) is no smaller than the right-hand side. We now show the inequality goes in the opposite direction as well. Note that there exist distinct $i^*, j^* \in [n]$ such that:

$$d_\infty^{\text{pair}}(M, M') = |M_{\{i^*,j^*\}}(i^*) - M'_{\{i^*,j^*\}}(i^*)|.$$

Now, if

$$|M_{\{i^*,j^*\}}(i^*) - M'_{\{i^*,j^*\}}(i^*)| = M'_{\{i^*,j^*\}}(i^*) - M_{\{i^*,j^*\}}(i^*),$$

then:

$$\begin{aligned} d_\infty^{\text{pair}}(M, M') &= M'_{\{i^*,j^*\}}(i^*) - M_{\{i^*,j^*\}}(i^*) \\ &\le \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} M'_{\{i,j\}}(i) - M_{\{i,j\}}(i) \end{aligned}$$

and hence:

$$d_\infty^{\text{pair}}(M, M') \leq \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} \text{val}^i(j).$$

Otherwise, we must have:

$$|M_{\{i^*,j^*\}}(i^*) - M'_{\{i^*,j^*\}}(i^*)| = M_{\{i^*,j^*\}}(i^*) - M'_{\{i^*,j^*\}}(i^*),$$

which implies:

$$\begin{aligned} d_\infty^{\text{pair}}(M, M') &= M_{\{i^*,j^*\}}(i^*) - M'_{\{i^*,j^*\}}(i^*) \\ &= 1 - M_{\{i^*,j^*\}}(j^*) - (1 - M'_{\{i^*,j^*\}}(j^*)) \\ &= M'_{\{i^*,j^*\}}(j^*) - M_{\{i^*,j^*\}}(j^*) \\ &\leq \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} M'_{\{i,j\}}(i) - M_{\{i,j\}}(i), \end{aligned}$$

giving once again:

$$d_\infty^{\text{pair}}(M, M') \leq \max_{i\in[n]} \max_{j\in[n]\setminus\{i\}} \text{val}^i(j). \qquad \square$$

In light of this lemma, we focus on the problem of finding, for any fixed $i \in [n]$:

$$\max_{j\in[n]\setminus\{i\}} \text{val}^i(j).$$

Let the weights of the items in M be given by $w_1, ..., w_n$ and the weights of the items in M' be given by $v_1, ..., v_n$. Fixed any $i \in [n]$ and $\alpha \in [0, 1]$, consider the problem:

$$\begin{aligned} \max_{j\in[n]\setminus\{i\}} \quad & \alpha - \frac{w_i}{w_i + w_j} \\ \text{s.t.} \quad & \frac{v_i}{v_i + v_j} \geq \alpha. \end{aligned} \qquad (\mathtt{P}(i, \alpha, M, M'))$$

Lemma D.2. *Let $i \in [n]$ and let j^* be a solution to:*

$$\max_{j\in[n]\setminus\{i\}} \text{val}^i(j).$$

Suppose α is such that $\frac{v_i}{v_i+v_{j^*}} - \varepsilon < \alpha \leq \frac{v_i}{v_i+v_{j^*}}$, and let $j_{\alpha,i}$ be an optimal solution to $\mathtt{P}(i, \alpha, M, M')$, then:

$$\text{val}^i(j_{\alpha,i}) \geq \text{val}^i(j^*) - \varepsilon.$$

Proof. We have:

$$\begin{aligned} \text{val}^i(j_{\alpha,i}) &= \frac{v_i}{v_i + v_{j_{\alpha,i}}} - \frac{w_i}{w_i + w_{j_{\alpha,i}}} \\ &\geq \alpha - \frac{w_i}{w_i + w_{j_{\alpha,i}}} \\ &\geq \alpha - \frac{w_i}{w_i + w_{j^*}} \\ &> \frac{v_i}{v_i + v_{j^*}} - \varepsilon - \frac{w_i}{w_i + w_{j^*}} \\ &= \text{val}^i(j^*) - \varepsilon, \end{aligned}$$

where the first inequality follows from feasibility of $j_{\alpha,i}$ in $\mathtt{P}(i, \alpha, M, M')$ (note that a feasible solution exists since by assumption j^* is feasible), the second inequality follows from optimality of $j_{\alpha,i}$ in $\mathtt{P}(i, \alpha, M, M')$, and the third one follows from the fact that $\alpha > \frac{v_i}{v_i+v_{j^*}} - \varepsilon$. $\square$

We now show that, instead of solving $\mathtt{P}(i, \alpha, M, M')$, we can restrict the items to a smaller set on which it is easier to solve the problem. We begin with the following observation about Algorithm 3.

Lemma D.3. *Let $((w_i)_{i\in[n]}, (v_i)_{i\in[n]})$ be two ordered sets of positive numbers, such that $v_1 \leq \cdots \leq v_n$, then* `Filter` *returns, in $O(n)$ time, an ordered subset $I \subseteq [n]$ such that:*

1. *For each* $i, j \in I$: $w_i \leq w_j \iff v_i \leq v_j \iff i \leq j$,
2. *For every* $j \in [n] \setminus I$ *there exists some* $j' \in I$ *such that* $w_{j'} \geq w_j$ *and* $v_{j'} \leq v_j$.

Proof. At each iteration of the loop, the value of j is increased by 1 and hence, throughout the algorithm, j takes on all the values from 1 to n exactly once. This immediately yields the bound on the running time. Moreover, for each of these values: the value is included in I if and only if $w_i < w_j$. Since $v_1 \leq \cdots \leq v_n$ by assumption, the two properties above hold. □

We also have the following.

Lemma D.4. *Let* $I \subseteq [n]$ *be the output of* `Filter`$((w_\ell)_{\ell\in[n]}, (v_\ell)_{\ell\in[n]})$. *Consider the problem* $\mathtt{R}(i, I, \alpha, M, M')$ *defined below:*

$$\max_{j \in I \setminus \{i\}} \quad w_j \qquad (\mathtt{R}(i, I, \alpha, M, M'))$$
$$\textit{s.t.} \quad v_j \leq \frac{1-\alpha}{\alpha} \cdot v_i.$$

With $\alpha > 0$*, any optimizer* j^* *of* $\mathtt{R}(i, I, \alpha, M, M')$ *is also an optimizer of* $\mathtt{P}(i, \alpha, M, M')$.

Proof. A little algebra shows that the optimizers of $\mathtt{P}(i, \alpha, M, M')$ are the same j's that solve:

$$\max_{j \in [n] \setminus \{i\}} \quad w_j \qquad (\mathtt{Q}(i, \alpha, M, M'))$$
$$\text{s.t.} \quad v_j \leq \frac{1-\alpha}{\alpha} \cdot v_i.$$

We now show that any optimizer of $\mathtt{R}(i, I, \alpha, M, M')$ is also an optimizer of $\mathtt{Q}(i, \alpha, M, M')$, and this will complete the proof. To do so, we just have to argue that for any value $j \in [n]$ such that $v_j \leq \frac{1-\alpha}{\alpha} v_i$ there exists some $j' \in I$ with $v_{j'} \leq \frac{1-\alpha}{\alpha} v_i$ and such that $w_{j'} \geq w_j$. This is trivially true if $j \in I$. On the other hand, if $j \notin I$ then j was removed from I by Algorithm 3, and hence, by Lemma D.3, there exists some $j' \in I$ such that $w_{j'} \geq w_j$ but $v_{j'} \leq v_j \leq \frac{1-\alpha}{\alpha} v_i$ as needed. □

We are now ready to prove that Algorithm 2 returns an approximately worst pair in $O(\frac{n \log n}{\varepsilon})$ time.

Proof of Theorem 2.3. We first argue about the approximation guarantees of the algorithm. The algorithm considers several possible pairs (i, j), and it keeps track of the pair with the largest value of $\mathrm{val}^i(j)$. It is therefore sufficient to prove that, at some point, the algorithm considers a pair $(\bar{i}, \bar{j})$ such that:

$$\mathrm{val}^{\bar{i}}(\bar{j}) \geq \max_{i \in [n]} \max_{j \in [n] \setminus \{i\}} \mathrm{val}^i(j) - \varepsilon.$$

We will in fact show a stronger statement, namely that, for each $i \in [n]$, the algorithm will consider a pair $(i, \bar{j})$ such that:

$$\mathrm{val}^i(\bar{j}) \geq \max_{j \in [n] \setminus \{i\}} \mathrm{val}^i(j) - \varepsilon.$$

Fix some $i \in [n]$ and consider the iteration of the outer **for** loop corresponding to that choice of i. Let $j^* \in [n] \setminus \{i\}$ be such that $\mathrm{val}^i(j^*) = \max_{j \in [n] \setminus \{i\}} \mathrm{val}^i(j)$. Note that $\frac{v_i}{v_i + v_{j^*}} \in (0, 1)$ and $\varepsilon_1 < \varepsilon$. Hence, on some iteration of the inner **for** loop, α is such that:

$$\frac{v_i}{v_i + v_{j^*}} - \varepsilon < \alpha \leq \frac{v_i}{v_i + v_{j^*}}.$$

Observe that, on that iteration, $\mathtt{R}(i, I, \alpha, M, M')$ is feasible and moreover, the binary search within Algorithm 2 (Line 8) finds the optimizer $j_{\alpha,i}$ of $\mathtt{R}(i, I, \alpha, M, M')$. Indeed, by Lemma D.3, the set I has the property that for any $\ell, k \in I$ with $\ell \leq k$ it holds that $v_\ell \leq v_k$ and $w_\ell \leq w_k$. Hence to maximize the value of w_j, one simply wants to maximize j subject to the constraint that $v_j \leq \frac{1-\alpha}{\alpha} \cdot v_i$, which is exactly what the binary search does.

By Lemma D.4, $j_{\alpha,i}$ is also an optimizer of $\mathtt{P}(i, \alpha, M, M')$ (note that this is true also for $\alpha = 0$ since in this case we have to take the maximum j in $I \setminus \{i\}$). Therefore, by Lemma D.2, this has the property that

$$\mathrm{val}^i(j_{\alpha,i}) \geq \max_{j \in [n] \setminus \{i\}} \mathrm{val}^i(j) - \varepsilon,$$

as needed.

For the time complexity, we simply note that

$$\varepsilon_1 = \left\lceil \frac{1}{\varepsilon} \right\rceil^{-1} \geq \left(\frac{1}{\varepsilon} + 1 \right)^{-1} = \frac{\varepsilon}{1+\varepsilon} \geq \frac{\varepsilon}{1+\frac{1}{3}} = \frac{3}{4}\varepsilon$$

and hence the body of the inner for loop is executed at most $O(\frac{n}{\varepsilon})$ times. Each time, the dominant cost is the binary search which takes time $O(\lceil \log_2 |I| \rceil) \leq O(\log n)$, and hence the algorithm takes at most $O(\frac{n}{\varepsilon} \log n)$ time. □

E Omitted Proofs from Section 2.3

In this section, we prove Theorem 2.4. In particular, we describe an algorithm which, given explicit access to the weights of two MNLs M and M', approximates the value of $d_1(M, M')$. For convenience, let us define:

$$d_{\mathrm{TV}}(M, M') := \max_{\substack{S \subseteq [n] \\ S \neq \varnothing}} \|M_S - M'_S\|_{\mathrm{TV}},$$

where $\|\cdot\|_{\mathrm{TV}}$ is the total variation distance. Since it holds that $\|M_S - M'_S\|_{\mathrm{TV}} = \frac{1}{2} \cdot \|M_S - M'_S\|_1$, we have $d_1(M, M') = 2 \cdot d_{\mathrm{TV}}(M, M')$. Here, we will show how to approximate d_{TV} within ε and this will immediately imply an approximation algorithm for d_1. By definition of total variation distance, it also holds that:

$$d_{\mathrm{TV}}(M, M') = \max_{\varnothing \subset T \subset S \subseteq [n]} \left| \frac{w(T)}{w(S)} - \frac{v(T)}{v(S)} \right|,$$

where $\{w_i\}_{i \in [n]}$ and $\{v_i\}_{i \in [n]}$ are the weights of the two MNLs.

Given a collection of numbers $\{v_i\}_{i \in [n]}$, and a subset $S \subseteq [n]$, we will write $v(S)$ for the value $v(S) := \sum_{i \in S} v_i$.

We begin by noting the following lemma:

Lemma E.1 (Weight Compression Lemma)**.** *There exists an algorithm which takes as input an MNL M supported on $[n]$ and a value $\varepsilon \in [0, 1/2]$, and, after making at most $O(n)$ arithmetic operations on the items of M, returns an MNL M' supported on $[n]$ with weights $w'_1, \ldots, w'_n$ such that:*

$$d_{\mathrm{TV}}(M, M') \leq 3\varepsilon$$

and:

$$\max_{i,j \in [n]} \frac{w'_i}{w'_j} \leq \left(\frac{n}{\varepsilon} \right)^{n-1}. \tag{3}$$

This lemma allows us to assume that the weights of the MNL we are considering are close to each other. We note that a version of this result appears implicitly in prior work (see e.g. the construction in the first part of the proof of Theorem 10 in the paper of Chierichetti et al. (2021)), we provide an explicit proof for completeness.

Proof. The algorithm works as follows. First, it sorts the weights $\{w_1, ..., w_n\}$ of the items of M in increasing order, so that $w_1 \leq w_2 \leq \cdots \leq w_n$. Then, it sets $w'_1 = w_1$, and for each $i \in [n] \setminus \{1\}$ it sets:

$$w'_i = w'_{i-1} \cdot \min \left\{ \frac{w_i}{w_{i-1}}, \frac{n}{\varepsilon} \right\}.$$

By construction, we have that, for every $i \in [n-1]$, $w'_i \leq w'_{i+1} \leq w'_i \cdot \frac{n}{\varepsilon}$, and hence:

$$\max_{i,j \in [n]} \frac{w'_i}{w'_j} = \frac{w'_n}{w'_1} = \prod_{i=1}^{n-1} \frac{w'_{i+1}}{w'_i} \leq \left(\frac{n}{\varepsilon} \right)^{n-1},$$

giving (3).

We now show that $d_{\mathrm{TV}}(M, M') \leq 3\varepsilon$. Consider any $\varnothing \subset S \subseteq [n]$. Let $S = \{s_1, \cdots, s_k\}$, and, for the sake of argument, relabel the items in S so that they have weights $w_{s_1} \leq \cdots \leq w_{s_k}$ in M and weights $w'_{s_1} \leq \cdots \leq w'_{s_k}$ in M', and rescale the weights so that $w_{s_k} = w'_{s_k}$.

Suppose that for every $i \in [k-1]$: $\frac{w_{s_{i+1}}}{w_{s_i}} \leq \frac{n}{\varepsilon}$. Under this assumption, by construction we have $w_{s_i} = w'_{s_i}$ for all $i \in [k]$, and hence $\|M_S - M'_S\|_1 = 0$.

Suppose the assumption does not hold. Then there exists some $i \in [k-1]$ such that $\frac{w_{s_{i+1}}}{w_{s_i}} > \frac{n}{\varepsilon}$. Let $i^* := \max\{i \in [k-1] \mid {}^{w_{s_{i+1}}}/_{w_{s_i}} > {}^{n}/_{\varepsilon}\}$. We can partition S into $S_l = \{s_1, ..., s_{i^*}\}$ (l stands for "low") and $S_h := \{s_{i^*+1}, ..., s_k\}$ (h stands for "high"). Intuitively, all the elements in S_h will have the same relative weighting in both M and M' by construction, while the elements in S_l will account for at most a ε-fraction of the total weight. In fact, by construction we have:

$$\frac{w_i}{w_j} = \frac{w'_i}{w'_j} \qquad \text{for all } i, j \in S_h, \tag{4}$$

as well as:

$$w_i \leq \frac{\varepsilon}{n} \cdot w_j \qquad \text{for all } i \in S_l \text{ and } j \in S_h, \tag{5}$$

and:

$$w'_i \leq \frac{\varepsilon}{n} \cdot w'_j \qquad \text{for all } i \in S_l \text{ and } j \in S_h. \tag{6}$$

Now, consider any subset $T \subseteq S$. We have:

$$\begin{aligned}
\frac{w(T)}{w(S)} &= \frac{w(T \cap S_h) + w(T \cap S_l)}{w(S)} \\
&= \frac{w(T \cap S_h)}{w(S)} + \frac{w(T \cap S_l)}{w(S)} \\
&\leq \frac{w(T \cap S_h)}{w(S)} + \frac{w(T \cap S_l)}{w_{s_k}} \\
&\overset{(5)}{\leq} \frac{w(T \cap S_h)}{w(S)} + |T \cap S_l| \cdot \frac{\varepsilon}{n} \\
&\leq \frac{w(T \cap S_h)}{w(S)} + \varepsilon \\
&\leq \frac{w(T \cap S_h)}{w(S_h)} + \varepsilon \\
&= \sum_{i \in T \cap S_h} \frac{1}{\sum_{j \in S_h} \frac{w_j}{w_i}} + \varepsilon \\
&\overset{(4)}{=} \sum_{i \in T \cap S_h} \frac{1}{\sum_{j \in S_h} \frac{w'_j}{w'_i}} + \varepsilon \\
&= \frac{w'(T \cap S_h)}{w'(S_h)} + \varepsilon \\
&\overset{(6)}{\leq} \frac{1}{(1-\varepsilon)} \cdot \frac{w'(T \cap S_h)}{w'(S)} + \varepsilon \\
&\leq \frac{1}{(1-\varepsilon)} \cdot \frac{w'(T)}{w'(S)} + \varepsilon \\
&\leq (1+2\varepsilon) \cdot \frac{w'(T)}{w'(S)} + \varepsilon \\
&\leq \frac{w'(T)}{w'(S)} + 3\varepsilon.
\end{aligned}$$

And hence, for all T, we also have:

$$\frac{w(T)}{w(S)} = 1 - \frac{w(S \setminus T)}{w(S)} \geq 1 - \frac{w'(S \setminus T)}{w'(S)} - 3\varepsilon = \frac{w'(T)}{w'(S)} - 3\varepsilon.$$

In particular, $\left|\frac{w(T)}{w(S)} - \frac{w'(T)}{w'(S)}\right| \leq 3\varepsilon$ for every choice of S and T, and hence $d_{\mathrm{TV}}(M, M') \leq 3\varepsilon$ as needed. □

We will also make use of the following result.

Lemma E.2. *Given two collections of positive real numbers $\{w_i\}_{i\in[n]}$ and $\{v_i\}_{i\in[n]}$ with $\min_i w_i = \min_i v_i = 1$, and positive real numbers $a, b, c, d \in \mathbb{R}_{>0}$ and $\varepsilon \in (0, 1/3)$ we consider the following problem:*

$$\begin{array}{rll} \text{find} & S, T \text{ with } \varnothing \subset T \subset S \subseteq [n] & (\mathrm{KS}(a,b,c,d)) \\ \textit{s.t.} & a \leq v(T) \leq a(1+\varepsilon) & \\ & b \leq w(S) \leq b(1+\varepsilon) & \\ & w(T) \leq c & \\ & v(S) \leq d. & \end{array}$$

and define an approximate solution as a pair of sets S and T with $\varnothing \subset T \subset S \subseteq [n]$ satisfying:

$$\begin{array}{l} a(1-\varepsilon) \leq v(T) \leq a(1+\varepsilon)^2 \\ b(1-\varepsilon) \leq w(S) \leq b(1+\varepsilon)^2 \\ w(T) \leq c(1+\varepsilon) \\ v(S) \leq d(1+\varepsilon). \end{array}$$

Then, there exists an algorithm that, given $\{w_i\}_{i\in[n]}$, $\{v_i\}_{i\in[n]}$, a, b, c, d, and ε satisfies the following conditions:

1. *(Soundness) The algorithm always either returns an approximate solution to KS(a, b, c, d) or $\perp$,*
2. *(Completeness) If KS(a, b, c, d) is feasible, i.e. if there is an exact solution, then the algorithm returns an approximate solution,*
3. *The algorithm can be implemented to run in* expected $n^{O\left(\frac{1}{\varepsilon^2}\right)}$ *time.*

Proof. For any element $i \in [n]$, we say that i is *heavy*, if it belongs to the set:

$$H := \left\{ i \in [n] \;\middle|\; \max\left\{\frac{v_i}{a}, \frac{w_i}{b}, \frac{w_i}{c}, \frac{v_i}{d}\right\} \geq \frac{\varepsilon^2}{200} \right\},$$

we say it is *S-heavy* if it belongs to the set:

$$H^S := \left\{ i \in [n] \;\middle|\; \max\left\{\frac{w_i}{b}, \frac{v_i}{d}\right\} \geq \frac{\varepsilon^2}{200} \right\},$$

and we say it is *light* otherwise.

We define the *heavy item projection* of a solution (S, T) to the optimization problem above as the pair $(S \cap H^S, T \cap H)$. We remark the following simple result.

Lemma E.3. *Let (S, T) be a feasible solution to* (KS(a, b, c, d)) *then $|S \cap H^S| \leq \frac{800}{\varepsilon^2}$ and $|T \cap H| \leq \frac{1600}{\varepsilon^2}$.*

Proof of Lemma E.3. We show the statement of the lemma by contradiction. Suppose $|S \cap H^S| > \frac{800}{\varepsilon^2}$. Let:

$$B_S = \left\{ i \in S \;\middle|\; \frac{w_i}{b} \geq \frac{\varepsilon^2}{200} \right\} \text{ and } D_S = \left\{ i \in S \;\middle|\; \frac{v_i}{d} \geq \frac{\varepsilon^2}{200} \right\}$$

and note that $S \cap H^S = B_S \cup D_S$. Since $|S \cap H^S| > \frac{800}{\varepsilon^2}$, by the pigeonhole principle, we must have that there exists some set $E \in \{B_S, D_S\}$ such that $|E| > \frac{400}{\varepsilon^2}$. Fix any such E, then either $E = B_S$ or $E = D_S$, we show that both of these possibilities imply a contradiction.

If $E = B_S$ then:

$$w(S) \geq w(B_S) > \frac{400}{\varepsilon^2} \cdot \frac{\varepsilon^2 b}{200} = 2b > b(1+\varepsilon).$$

If $E = D_S$ then:

$$v(S) \geq v(D_S) > \frac{400}{\varepsilon^2} \cdot \frac{\varepsilon^2 d}{200} = 2d > d,$$

and hence (S, T) is not a feasible solution to KS(a, b, c, d) in any of these cases, giving a contradiction. This completes the proof that $|S \cap H^S| \leq \frac{800}{\varepsilon^2}$.

We prove that $|T \cap H| \leq \frac{1600}{\varepsilon^2}$ in an analogous way. Suppose that $|T \cap H| > \frac{1600}{\varepsilon^2}$. Then, define:

$$A_T := \left\{ i \in T \;\middle|\; \frac{v_i}{a} \geq \frac{\varepsilon^2}{200} \right\}, \quad B_T := \left\{ i \in T \;\middle|\; \frac{w_i}{b} \geq \frac{\varepsilon^2}{200} \right\},$$

$$C_T := \left\{ i \in T \;\middle|\; \frac{w_i}{c} \geq \frac{\varepsilon^2}{200} \right\}, \quad D_T := \left\{ i \in T \;\middle|\; \frac{v_i}{d} \geq \frac{\varepsilon^2}{200} \right\}.$$

Note that $T \cap H = A_T \cup B_T \cup C_T \cup D_T$ and hence, by the pigeonhole principle, there exists some $E \in \{A_T, B_T, C_T, D_T\}$ such that $|E| > \frac{400}{\varepsilon^2}$. Fix any such E. We then have four cases, each of which implies a contradiction:

Case 1 $E = A_T$. In this case, we have:

$$v(T) \geq v(A_T) \geq \frac{400}{\varepsilon^2} \cdot \frac{\varepsilon^2 a}{200} = 2a > a(1+\varepsilon).$$

Case 2 $E = B_T$. In this case, we have:

$$w(S) \geq w(T) \geq w(B_T) \geq \frac{400}{\varepsilon^2} \cdot \frac{\varepsilon^2 b}{200} = 2b > b(1+\varepsilon).$$

Case 3 $E = C_T$. In this case, we have:

$$w(T) \geq w(C_T) \geq \frac{400}{\varepsilon^2} \cdot \frac{\varepsilon^2 c}{200} = 2c > c.$$

Case 4 $E = D_T$. In this case, we have:

$$v(S) \geq v(T) \geq v(D_T) \geq \frac{400}{\varepsilon^2} \cdot \frac{\varepsilon^2 d}{200} = 2d > d.$$

Note that (S, T) is not a feasible solution in any of these cases, giving a contradiction. Hence $|T \cap H| \leq \frac{1600}{\varepsilon^2}$, completing the proof of Lemma E.3. □

In turn, this result allows us to prove the following.

Lemma E.4. *Let $\mathcal{HIP}$ be the set of heavy item projections induced by feasible solutions to the problem above, i.e.:*

$$\mathcal{HIP} = \{(S \cap H^S, T \cap H) \mid (S, T) \text{ is a feasible solution to KS}(a, b, c, d)\}.$$

There exists an algorithm that in time $n^{O\left(\frac{1}{\varepsilon^2}\right)}$ enumerates all the elements of a superset of $\mathcal{HIP}$.

Proof of Lemma E.4. By Lemma E.3, we have:

$$\mathcal{HIP} \subseteq \left\{ (A_1, A_2) \,\middle|\, A_i \subseteq [n], |A_i| \leq \frac{1600}{\varepsilon^2} : i \in \{1,2\} \right\}$$

where the second set can be enumerated efficiently, and it has cardinality:

$$\begin{aligned}
&\left| \left\{ (A_1, A_2) \,\middle|\, A_i \subseteq [n], |A_i| \leq \frac{1600}{\varepsilon^2} : i \in \{1,2\} \right\} \right| \\
&\leq \left(\sum_{i=1}^{\lfloor 1600/\varepsilon^2 \rfloor} \binom{n}{i} \right)^2 \\
&\leq (n+1)^{\frac{3200}{\varepsilon^2}} \\
&= n^{O\left(\frac{1}{\varepsilon^2}\right)},
\end{aligned}$$

where we used that $\sum_{i=0}^{k} n^i \leq (n+1)^k$ for $k \leq n$. □

Note in particular, that this implies:

$$|\mathcal{HIP}| \leq n^{O\left(\frac{1}{\varepsilon^2}\right)}.$$

Intuitively, we will want to guess the heavy element projection of a feasible solution, by enumerating all the possibilities. Given a pair $P = (S_h, T_h)$, representing one such possibility, we can then consider the program:

$$\begin{aligned}
\text{find} \quad & x, y \in [0,1]^n \text{ with } y \leq x && \text{(KS-LP}(a,b,c,d,P)\text{)} \\
\text{s.t.} \quad & a \leq \sum_{i \in [n]} y_i v_i \leq a(1+\varepsilon) \\
& b \leq \sum_{i \in [n]} x_i w_i \leq b(1+\varepsilon) \\
& \sum_{i \in [n]} y_i w_i \leq c \\
& \sum_{i \in [n]} x_i v_i \leq d \\
& x_i = 1, \quad \forall i \in S_h \\
& y_i = 1, \quad \forall i \in T_h \\
& x_i = 0, \quad \forall i \in H^S \setminus S_h \\
& y_i = 0, \quad \forall i \in H \setminus T_h.
\end{aligned}$$

This linear program represents a continuous relaxation of the problem of extending the candidate projection P to a feasible solution (S, T) for the original program KS(a, b, c, d). We now show that if KS-LP(a, b, c, d, P) is feasible, any feasible solution can be rounded to an approximately feasible solution $(\hat{S}, \hat{T})$ to KS(a, b, c, d).

Lemma E.5. *If the program KS-LP(a, b, c, d, P) is feasible, then, with expected running time $O(n)$, one can round a solution to the program to two sets $\hat{S}$ and $\hat{T}$ with the property that:*

$$\begin{aligned}
&(1-\varepsilon)a \leq v(\hat{T}) \leq (1+\varepsilon)^2 a \\
&(1-\varepsilon)b \leq w(\hat{S}) \leq (1+\varepsilon)^2 b \\
&w(\hat{T}) \leq (1+\varepsilon)c \\
&v(\hat{S}) \leq (1+\varepsilon)d.
\end{aligned} \tag{7}$$

Proof of Lemma E.5. Consider the random sets $\hat{S}$ and $\hat{T}$, obtained by the following coupled process: we sample $z_1, \ldots, z_n \sim \mathrm{Uniform}([0,1])$ independently, for each $i \in [n]$, we let:

$$X_i = \mathbb{1}\{z_i \leq x_i\} \text{ and } Y_i = \mathbb{1}\{z_i \leq y_i\},$$

and define:

$$\hat{S} = \{i \in [n] \mid X_i = 1\} \text{ and } \hat{T} = \{i \in [n] \mid Y_i = 1\}.$$

We then have:

$$\mathbb{E}\left[v(\hat{T})\right] = \mathbb{E}\left[\sum_{i\in[n]} Y_i v_i\right] = \sum_{i\in[n]} \mathbb{E}\left[Y_i\right] v_i = \sum_{i\in[n]} y_i v_i \in (a, a(1+\varepsilon))$$

and:

$$\begin{aligned}
\mathrm{Var}[v(\hat{T})] &= \mathrm{Var}\left[\sum_{i\in[n]} Y_i v_i\right] \\
&= \sum_{i\in[n]} \mathrm{Var}\left[Y_i v_i\right] \\
&= \sum_{i\notin H} \mathrm{Var}\left[Y_i v_i\right] + \cancelto{0}{\sum_{i\in H} \mathrm{Var}\left[Y_i v_i\right]} \\
&= \sum_{i\notin H} \mathrm{Var}\left[Y_i\right] v_i^2 \\
&\leq \sum_{i\notin H} y_i \cdot v_i \cdot v_i \\
&\leq \frac{\varepsilon^2 a}{200} \sum_{i\notin H} y_i \cdot v_i \\
&\leq \frac{(1+\varepsilon)\varepsilon^2 a^2}{200}
\end{aligned}$$

where the second equality follows by independence, and the second inequality uses the fact that i is a light element. Hence, by Chebyshev's Inequality:

$$\begin{aligned}
\Pr\left[v(\hat{T}) \notin \left(a(1-\varepsilon), a(1+\varepsilon)^2\right)\right] &\leq \Pr\left[v(\hat{T}) \notin \left(a - 10\sqrt{\frac{(1+\varepsilon)}{200}}\varepsilon a, a(1+\varepsilon) + 10\sqrt{\frac{(1+\varepsilon)}{200}}\varepsilon a\right)\right] \\
&\leq \Pr\left[v(\hat{T}) \notin \left(a - 10\sqrt{\mathrm{Var}[v(\hat{T})]}, a(1+\varepsilon) + 10\sqrt{\mathrm{Var}[v(\hat{T})]}\right)\right] \\
&\leq \Pr\left[v(\hat{T}) \notin \left(\mathbb{E}\left[v(\hat{T})\right] - 10\sqrt{\mathrm{Var}[v(\hat{T})]}, \mathbb{E}\left[v(\hat{T})\right] + 10\sqrt{\mathrm{Var}[v(\hat{T})]}\right)\right] \\
&= \Pr\left[\left|v(\hat{T}) - \mathbb{E}\left[v(\hat{T})\right]\right| > 10\sqrt{\mathrm{Var}[v(\hat{T})]}\right] \\
&\leq \frac{1}{100}.
\end{aligned}$$

We proceed in an analogous way to obtain similar bounds for the other constraints. Starting from the b constraint, we have:

$$\mathbb{E}\left[w(\hat{S})\right] = \mathbb{E}\left[\sum_{i\in[n]} X_i w_i\right] \in (b, b(1+\varepsilon))$$

and:

$$\mathrm{Var}[w(\hat{S})] = \mathrm{Var}\left[\sum_{i\in[n]} X_i w_i\right] = \sum_{i\notin H^S} \mathrm{Var}\left[X_i\right] w_i^2 \leq \frac{\varepsilon^2 b}{200} \sum_{i\notin H^S} x_i \cdot w_i \leq \frac{(1+\varepsilon)\varepsilon^2 b^2}{200}$$

where we are using the fact that i is not S-heavy. Hence, by Chebyshev's Inequality:

$$\Pr\left[w(\hat{S}) \notin \left(b(1-\varepsilon), b(1+\varepsilon)^2\right)\right] \leq \Pr\left[w(\hat{S}) \notin \left(b - 10\sqrt{\mathrm{Var}[w(\hat{S})]}, b(1+\varepsilon) + 10\sqrt{\mathrm{Var}[w(\hat{S})]}\right)\right] \leq \frac{1}{100}.$$

We can apply the same reasoning to the c constraint, and obtain:

$$\mathbb{E}\left[w(\hat{T})\right] = \mathbb{E}\left[\sum_{i\in[n]} Y_i w_i\right] \leq c$$

and:

$$\mathrm{Var}[w(\hat{T})] = \mathrm{Var}\left[\sum_{i\in[n]} Y_i w_i\right] = \sum_{i\notin H} \mathrm{Var}\left[Y_i\right] w_i^2 \leq \frac{\varepsilon^2 c}{200} \sum_{i\notin H} y_i \cdot w_i \leq \frac{\varepsilon^2 c^2}{200},$$

where we are using the fact that every $i \notin H$ is a light element. Hence, by Chebyshev's Inequality:

$$\Pr\left[w(\hat{T}) \geq c(1+\varepsilon)\right] \leq \Pr\left[w(\hat{T}) - \mathbb{E}\left[w(\hat{T})\right] \geq \varepsilon c\right] \leq \frac{\mathrm{Var}[w(\hat{T})]}{(\varepsilon c)^2} \leq \frac{1}{200} \leq \frac{1}{100}.$$

Finally, we look at the d constraint and obtain:

$$\mathbb{E}\left[v(\hat{S})\right] = \mathbb{E}\left[\sum_{i\in[n]} X_i v_i\right] \leq d$$

and:

$$\mathrm{Var}[v(\hat{S})] = \mathrm{Var}\left[\sum_{i\in[n]} X_i v_i\right] = \sum_{i\notin H^S} \mathrm{Var}\left[X_i\right] v_i^2 \leq \frac{\varepsilon^2 d}{200} \sum_{i\notin H^S} x_i \cdot v_i \leq \frac{\varepsilon^2 d^2}{200},$$

where we are using the fact that every i in the sum is not S-heavy. Hence, by Chebyshev's Inequality:

$$\Pr\left[v(\hat{S}) \geq d(1+\varepsilon)\right] \leq \Pr\left[v(\hat{S}) - \mathbb{E}\left[v(\hat{S})\right] \geq \varepsilon d\right] \leq \frac{\mathrm{Var}[v(\hat{S})]}{(\varepsilon d)^2} \leq \frac{1}{200} \leq \frac{1}{100}.$$

Hence, by the union bound, with probability at least $\frac{24}{25}$, $(\hat{S}, \hat{T})$ is an approximately feasible solution to KS(a, b, c, d). We repeat this algorithm multiple times until we find an approximate feasible solution. In expectation, we will repeat the process $25/24$ times, for an expected $O(n)$ running time. □

Now, suppose there exists a feasible solution (S, T) to KS(a, b, c, d), then if we consider $P = (S_h, T_h)$ (the heavy element projection of (S, T)) the LP KS-LP(a, b, c, d, P) is feasible and hence, by Lemma E.5 we can obtain sets $\hat{S}, \hat{T} \subseteq [n] \setminus H$ with the guarantees of Equation (7). Hence, to guarantee the results in the statement of Lemma E.2 it is sufficient to try all the possible values of $P = (S_h, T_h)$ (of which there are only $n^{O(\varepsilon^{-2})}$ by Lemma E.4) and for each of them solve KS-LP(a, b, c, d, P). If KS(a, b, c, d) is feasible, then KS-LP(a, b, c, d, P) will be feasible for a choice of P that is the heavy element projection of a feasible solution to KS(a, b, c, d) and for that choice, we will find the wanted solution and return it. If no such choice exists, the algorithm will return $\perp$.

We now compute the running time of the algorithm. The algorithm tries $n^{O(1/\varepsilon^2)}$ candidate sets (S_h, T_h) and for each of them solves a linear program with $O(1)$ constraints and $O(n)$ variables for a running time of $n^{O(1)}$ using the ellipsoid method. Finally, when the algorithm finds a feasible solution to KS-LP(a, b, c, d, P) it runs the rounding algorithm guaranteed by Lemma E.5 with an expected $O(n)$ cost. Thus, the total expected running time is $n^{O(1/\varepsilon^2)} \cdot n^{O(1)} = n^{O(1/\varepsilon^2)}$. This concludes the proof of Lemma E.2. □

We can now prove the main theorem. Specifically, we will prove that Algorithm 4 returns an ε-additive approximation to $d_{\mathrm{TV}}(M_1, M_2)$. By running this algorithm with $\varepsilon/2$, we immediately obtain the guarantee for d_1 required by Theorem 2.4.

Algorithm 4 Estimate-$d_{\mathrm{TV}}(M_1, M_2, \varepsilon)$

Input: The weights $\{w_i\}_{i\in[n]}$ of an MNL M_1, the weights $\{v_i\}_{i\in[n]}$ of an MNL M_2, an approximation parameter $\varepsilon \in (0, 1/3)$.
Output: A slate S that is close to witnessing $d_{\mathrm{TV}}(M_1, M_2)$.
$\varepsilon_1 = \varepsilon/12$, $\varepsilon_2 = \varepsilon/30$
$\hat{d} = 0$
Rescale the weights of $\{v\}_{i\in[n]}$ and $\{w\}_{i\in[n]}$ using the algorithm guaranteed by Lemma E.1 with accuracy parameter ε_1, and obtain weights v' and w' corresponding to MNLs M_1' and M_2'
Rescale the values of $\{v'\}_{i\in[n]}$ and $\{w'\}_{i\in[n]}$, so that: $\min_{i\in[n]} v_i' = \min_{i\in[n]} w_i' = 1$.
$A = \left\{(1+\varepsilon_2)^i \mid i = 0, 1, \ldots, \left\lceil \log_{(1+\varepsilon_2)} \left(\frac{n}{\varepsilon_1}\right)^n \right\rceil \right\}$
for $(a, b, c, d) \in A^4$ **do**
    Run the algorithm guaranteed by Lemma E.2 with input $\{w_i'\}_{i\in[n]}$, $\{v_i'\}_{i\in[n]}$, (a, b, c, d), and ε_2. If the algorithm finds a solution (S, T), then:
    if $\frac{v'(T)}{v'(S)} - \frac{w'(T)}{w'(S)} > \hat{d}$ **then**
        $\hat{d} = \frac{v'(T)}{v'(S)} - \frac{w'(T)}{w'(S)}$
        $(S^*, T^*) = (S, T)$
    end if
end for
return $\hat{d}, (S^*, T^*)$

Theorem E.6. *There exists an algorithm which, on input two sets of weights $v_1, \ldots, v_n$, and $w_1, \ldots, w_n$ for two MNLs M_1 and M_2 respectively, as well as an accuracy parameter $\varepsilon \in (0, 1/3)$, finds, in* expected *time $n^{O(1/\varepsilon^2)}$, subsets $T \subseteq S \subseteq [n]$ such that:*

$$\left|\frac{w(T)}{w(S)} - \frac{v(T)}{v(S)}\right| \geq d_{\mathrm{TV}}(M_1, M_2) - \varepsilon.$$

Proof. Let $\varepsilon_3 = \varepsilon/2$. We will show that Algorithm 4 obtains a close estimate $\hat{d}$ for the value of $d_{\mathrm{TV}}(M_1', M_2')$, i.e. that:

$$|\hat{d} - d_{\mathrm{TV}}(M_1', M_2')| \leq \varepsilon_3.$$

By Lemma E.1, we will then have:

$$\begin{aligned}
|\hat{d} - d_{\mathrm{TV}}(M_1, M_2)| &= |\hat{d} - d_{\mathrm{TV}}(M_1', M_2') + d_{\mathrm{TV}}(M_1', M_2') - d_{\mathrm{TV}}(M_1, M_2)| \\
&\leq |\hat{d} - d_{\mathrm{TV}}(M_1', M_2')| + |d_{\mathrm{TV}}(M_1', M_2') - d_{\mathrm{TV}}(M_1, M_2)| \\
&\leq |\hat{d} - d_{\mathrm{TV}}(M_1', M_2')| + |d_{\mathrm{TV}}(M_1', M_2') - d_{\mathrm{TV}}(M_1, M_2')| + |d_{\mathrm{TV}}(M_1, M_2') - d_{\mathrm{TV}}(M_1, M_2)| \\
&\leq |\hat{d} - d_{\mathrm{TV}}(M_1', M_2')| + |d_{\mathrm{TV}}(M_1', M_1)| + |d_{\mathrm{TV}}(M_2', M_2)| \\
&\leq |\hat{d} - d_{\mathrm{TV}}(M_1', M_2')| + 6\varepsilon_1 \\
&\leq \varepsilon_3 + 6\varepsilon_1 \leq \varepsilon
\end{aligned}$$

giving the wanted approximation guarantees.

We note that there exist T^* and S^* with $\varnothing \subset T^* \subset S^* \subseteq [n]$ such that:

$$d_{\mathrm{TV}}(M_1', M_2') = \frac{v'(T^*)}{v'(S^*)} - \frac{w'(T^*)}{w'(S^*)}.$$

The algorithm will obtain a collection of guesses for the solution (S, T) and keep the guess that maximizes:

$$\frac{v'(T)}{v'(S)} - \frac{w'(T)}{w'(S)}.$$

We make use of the following key result

Lemma E.7. *There exists one collection of values (a, b, c, d) considered in the **for** loop of the algorithm such that, when the algorithm in Lemma E.2 is run with input $\{w'_i\}_{i\in[n]}$, $\{v'_i\}_{i\in[n]}$, (a, b, c, d) and ε_2 it returns a solution S, T such that:*

$$\frac{v'(T)}{v'(S)} - \frac{w'(T)}{w'(S)} \geq \left(\frac{v'(T^*)}{v'(S^*)} - \frac{w'(T^*)}{w'(S^*)}\right) - \varepsilon_3.$$

Note that Lemma E.7 immediately implies Theorem 2.4, since Algorithm 4 returns the best guess it finds out of all the attempts. We therefore simply have to prove Lemma E.7, which we now do.

Proof of Lemma E.7. Since we rescale v' and w' so that their minimum is 1, we have that $v'(T^*), v'(S^*), w'(T^*), w'(S^*) \leq \left(\frac{n}{\varepsilon_1}\right)^n$. Then, there is at least one iteration of the for loop in which:

$$a \leq v'(T^*) \leq (1+\varepsilon_2)a, \qquad b \leq w'(S^*) \leq (1+\varepsilon_2)b,$$
$$\frac{c}{1+\varepsilon_2} \leq w'(T^*) \leq c, \qquad \frac{d}{(1+\varepsilon_2)} \leq v'(S^*) \leq d.$$

In particular, we have that (S^*, T^*) is a feasible solution to KS(a, b, c, d) with ε_2. Therefore, Lemma E.2 guarantees that on that iteration, the algorithm will find a guess (S, T) and that this will satisfy:

$$a(1-\varepsilon_2) \leq v'(T) \leq a(1+\varepsilon_2)^2$$
$$b(1-\varepsilon_2) \leq w'(S) \leq b(1+\varepsilon_2)^2$$
$$w'(T) \leq c(1+\varepsilon_2)$$
$$v'(S) \leq d(1+\varepsilon_2)$$

Hence the pair (S, T) satisfies:

$$\begin{aligned}
\frac{v'(T)}{v'(S)} - \frac{w'(T)}{w'(S)} &\geq \frac{1-\varepsilon_2}{1+\varepsilon_2}\cdot\frac{a}{d} - \frac{1+\varepsilon_2}{1-\varepsilon_2}\cdot\frac{c}{b} \\
&\geq \frac{1-\varepsilon_2}{(1+\varepsilon_2)^3}\cdot\frac{v'(T^*)}{v'(S^*)} - \frac{(1+\varepsilon_2)^3}{1-\varepsilon_2}\cdot\frac{w'(T^*)}{w'(S^*)} \\
&\geq (1-\varepsilon_2)(1-3\varepsilon_2)\cdot\frac{v'(T^*)}{v'(S^*)} - (1+6\varepsilon_2)(1+2\varepsilon_2)\cdot\frac{w'(T^*)}{w'(S^*)} \\
&\geq (1-6\varepsilon_2)\cdot\frac{v'(T^*)}{v'(S^*)} - (1+9\varepsilon_2)\cdot\frac{w'(T^*)}{w'(S^*)} \\
&\geq \frac{v'(T^*)}{v'(S^*)} - \frac{w'(T^*)}{w'(S^*)} - 6\varepsilon_2 - 9\varepsilon_2 \\
&\geq \frac{v'(T^*)}{v'(S^*)} - \frac{w'(T^*)}{w'(S^*)} - \varepsilon_3,
\end{aligned}$$

concluding the proof of Lemma E.7. □

We conclude the proof of Theorem 2.4 by analyzing the time complexity of Algorithm 4. The subroutine at Line 9 gets called $|A|^4$ times. Note that:

$$\begin{aligned}
|A| &\leq 1 + n\cdot\log_{1+\varepsilon_2}(n/\varepsilon_1) = 1 + n\cdot\frac{\ln(n/\varepsilon_1)}{\ln(1+\varepsilon_2)} \\
&\leq 1 + \frac{2\cdot n\cdot\ln(n/\varepsilon_1)}{\varepsilon_2} = O\left(\frac{n\log(n/\varepsilon)}{\varepsilon}\right),
\end{aligned}$$

where we used that $\ln(1+x) \geq x/2$ for $x \in (0, 1)$. Since each call to the subroutine costs $n^{O(1/\varepsilon^2)}$ in expectation, we have a total expected running time of

$$O\left(\frac{n^4\log^4(n/\varepsilon)}{\varepsilon^4}\right)\cdot n^{O(1/\varepsilon^2)} = n^{O(1/\varepsilon^2)},$$

concluding the proof. □

Algorithm 5 `UpperBound-`$d_1(M, M', (R_i)_{i\in[N]}, (\mathcal{S}_i)_{i\in[N]})$

Input: The weights $\{w_i\}_{i\in[n]}$ of an MNL M, the weights $\{v_i\}_{i\in[n]}$ of an MNL M', disjoint non-empty subsets $(R_1, \ldots, R_N)$ such that $\cup_{i\in[N]} R_i = [n]$, and families $(\mathcal{S}_1, \ldots, \mathcal{S}_N)$ such that for each $i \in [N]$, $\mathcal{S}_i = \{(H_j, T_j)\}_j$ where for each j: (i) $R_i \subseteq H_j$, (ii) $H_j \cup T_j = \cup_{k=i}^{N} R_k$, and (iii) $H_j \cap T_j = \varnothing$.
Output: A value $\varepsilon \in [0, 2]$ such that $d_1(M, M') \le \varepsilon$.
$\varepsilon = 0$
for $i = 1, \ldots, N$ **do**
  $\varepsilon' = 2$
  for $(H, T) \in \mathcal{S}_i$ **do**
    $m = \min_{r\in R_i}\{w_r\}$, $m' = \min_{r\in R_i}\{v_r\}$
    $\varepsilon_1 = \frac{1}{m}\sum_{t\in T} w_t$, $\varepsilon_2 = \frac{1}{m'}\sum_{t\in T} v_t$
    Let $\mu \ge 1$ be such that for any $u, u' \in H$: $\frac{1}{\mu}\cdot\frac{w_u}{w_{u'}} \le \frac{v_u}{v_{u'}} \le \mu\cdot\frac{w_u}{w_{u'}}$
    $\psi_1 = \max\left\{1 - \frac{1}{\mu(1+\varepsilon_2)}, \mu(1+\varepsilon_1) - 1\right\}$; $\psi_2 = \max\left\{1 - \frac{1}{\mu(1+\varepsilon_1)}, \mu(1+\varepsilon_2) - 1\right\}$
    $\psi_3 = \max\left\{1 - \frac{1}{\mu(1+\varepsilon_2)}, 1 - \frac{1}{\mu(1+\varepsilon_1)}\right\}$; $\psi_3 = \psi_3 \cdot \min\{2, \mu(1 + \min\{\varepsilon_1, \varepsilon_2\})\}$
    $\varepsilon_3 = \min\{\psi_1, \psi_2, \psi_3\}$
    $\varepsilon' = \min\{\varepsilon', \varepsilon_1 + \varepsilon_2 + \varepsilon_3\}$
  end for
  $\varepsilon = \max\{\varepsilon, \varepsilon'\}$
end for
return ε

F Omitted Proofs from Section 2.4: Efficient Upper Bound on d_1

In this section, we provide more details on the algorithm described in Section 2.4, that obtains a valid upper bound on d_1. Specifically, we show that Algorithm 5 provides a valid upper bound to $d_1(M, M')$.

Theorem F.1. *For MNLs M, M' over $[n]$, let ε be the output of Algorithm 5. Then, $d_1(M, M') \le \varepsilon$.*

Proof. Fix any slate $S \subseteq [n]$, $|S| \ge 2$. It is sufficient to show that for any such choice, we have $\|M_S - M'_S\|_1 \le \varepsilon$. Let i be the smallest index such that $R_i \cap S \ne \varnothing$. Let us consider the i^{th} iteration of the outer for loop. Define $\mathcal{R} = \cup_{j=i}^{N} R_j$. Consider any pair (H, T) such that $H \cup T = \mathcal{R}$, $H \cap T = \varnothing$, and $R_i \subseteq H$. Consider any $s \in S \cap H$, and any $s^\star \in S \cap R_i \subseteq S \cap H$. Note that $S \setminus H \subseteq T$, thus,

$$\sum_{u\in S\setminus H} \frac{w_u}{w_s} \le \frac{1}{w_s}\cdot\sum_{u\in T} w_u = \frac{\varepsilon_1\cdot m}{w_s} \le \frac{\varepsilon_1\cdot w_{s^\star}}{w_s} \le \varepsilon_1\cdot\sum_{u\in S\cap H}\frac{w_u}{w_s} \le \varepsilon_1\cdot\sum_{u\in S}\frac{w_u}{w_s} \quad \text{and,} \tag{8}$$

$$\sum_{u\in S\setminus H} \frac{v_u}{v_s} \le \frac{1}{v_s}\cdot\sum_{u\in T} v_u = \frac{\varepsilon_2\cdot m'}{v_s} \le \frac{\varepsilon_2\cdot v_{s^\star}}{v_s} \le \varepsilon_2\cdot\sum_{u\in S\cap H}\frac{v_u}{v_s} \le \varepsilon_2\cdot\sum_{u\in S}\frac{v_u}{v_s}. \tag{9}$$

By the previous equations, we have that:

$$M'_S(s) = \frac{1}{\sum_{u\in S\cap H}\frac{v_u}{v_s} + \sum_{u\in S\setminus H}\frac{v_u}{v_s}} \overset{Equation\ (9)}{\ge} \frac{1}{(1+\varepsilon_2)\sum_{u\in S\cap H}\frac{v_u}{v_s}}$$
$$\overset{\text{Line (9)}}{\ge} \frac{1}{\mu(1+\varepsilon_2)\sum_{u\in S\cap H}\frac{w_u}{w_s}} = \frac{1}{\mu(1+\varepsilon_2)}\cdot M_{S\cap H}(s) \ge \frac{1}{\mu(1+\varepsilon_2)}\cdot M_S(s),$$

and a symmetric argument where we use Equation (8) shows:

$$M_S(s) \ge \frac{1}{(1+\varepsilon_1)\mu}\cdot M'_S(s).$$

Putting things together, we have:

$$\sum_{s\in S\cap H} |M_S(s) - M'_S(s)| \le \max\left\{1 - \frac{1}{\mu(1+\varepsilon_2)}, (1+\varepsilon_1)\mu - 1\right\}\cdot\sum_{s\in S\cap H} M_S(s) \le \psi_1,$$

$$\sum_{s\in S\cap H}|M_S(s)-M'_S(s)|\le \max\left\{1-\frac{1}{\mu(1+\varepsilon_1)},(1+\varepsilon_2)\mu-1\right\}\cdot\sum_{s\in S\cap H}M'_S(s)\le\psi_2,$$

$$\begin{aligned}\sum_{s\in S\cap H}|M_S(s)-M'_S(s)|&\le \max\left\{1-\frac{1}{\mu(1+\varepsilon_1)},1-\frac{1}{\mu(1+\varepsilon_2)}\right\}\cdot\sum_{s\in S\cap H}\max\left\{M'_S(s),M_S(s)\right\}\\&\le \max\left\{1-\frac{1}{\mu(1+\varepsilon_1)},1-\frac{1}{\mu(1+\varepsilon_2)}\right\}\cdot\min\{2,\mu(1+\varepsilon_1),\mu(1+\varepsilon_2)\}\le\psi_3.\end{aligned}$$

We finally obtain:

$$\begin{aligned}\|M_S-M'_S\|_1&=\sum_{s\in S\cap H}|M_S(s)-M'_S(s)|+\sum_{s\in S\cap T}|M_S(s)-M'_S(s)|\\&\le\min\{\psi_1,\psi_2,\psi_3\}+\sum_{s\in S\cap T}(|M_S(s)|+|M'_S(s)|)\\&\le\varepsilon_3+\varepsilon_1+\varepsilon_2.\end{aligned}$$

Since this holds for each pair (H,T) we must also have $\|M_S-M'_S\|_1\le\varepsilon'\le\varepsilon$. □

We also show that Algorithm 5 can be easily adapted to provide an upper bound for d_∞ rather than d_1:

Theorem F.2. *For any MNLs M,M' over $[n]$, let ε be the output of Algorithm 5 after changing Line (13) to $\varepsilon'=\min\{1,\varepsilon',\max\{\varepsilon_3,\varepsilon_4,\varepsilon_5\}\}$, where $\varepsilon_4=\frac{1}{m}\max_{t\in T}\{w_t\}$, and $\varepsilon_5=\frac{1}{m'}\max_{t\in T}\{v_t\}$. Then, $d_\infty(M,M')\le\varepsilon$.*

Proof. Consider any $s\in S\subseteq[n]$, $|S|\ge 2$. Let i be the smallest index such that $R_i\cap S\neq\varnothing$ and consider the i^{th} iteration of the outer for loop. Define $\mathcal{R}=\cup_{j=i}^N R_j$. Consider any pair (H,T) such that $H\cup T=\mathcal{R}$, $H\cap T=\varnothing$, and $R_i\subseteq H$. If $s\in S\cap H$, then by following the same proof of Theorem F.1 we have that:

$$|M_S(s)-M'_S(s)|\le\varepsilon_3\cdot M_S(s)\le\varepsilon'.$$

If instead $s\in S\cap T$, we have that:

$$|M_S(s)-M'_S(s)|\le\max\{M_S(s),M'_S(s)\}\le\max\left\{\frac{w_s}{m},\frac{v_s}{m'}\right\}\le\max\{\varepsilon_4,\varepsilon_5\}\le\varepsilon'.\qquad\square$$

Another way to obtain an upper bound on d_∞ is to observe that $d_\infty(M,M')\le\frac{1}{2}\cdot d_1(M,M')$. Thus, we can take the minimum between the bound obtained via Theorem F.2 and half the bound obtained via Theorem F.1.

Observe that this algorithm returns a valid upper bound for any partition $(R_1,\dots,R_N)$ of $[n]$ and any families of pairs $(\mathcal{S}_1,\dots,\mathcal{S}_N)$. However, the complexity of the algorithm and the quality of the upper bound found depend on how we choose these sets. We now describe how we perform this choice heuristically in practice.

The idea behind the algorithm is to find an ordering of the items $(R_1,\dots,R_N)$ so that, for any fixed $i\in[N]$, the items starting from R_i can be partitioned into a "head" and a "tail" such that the items in the head $\{w_j\}_{j\in H}$ and $\{v_j\}_{j\in H}$ are close to each other (Line 9), while the items in the tail are extremely small compared to those in $R_i\subseteq H$ (Line 8).

Given this intuition, we build the partition by clustering close-by items as follows. We sort the items $\{w_i\}_{i\in[n]}$ of M so that $w_1\ge w_2\ge\cdots\ge w_n$, then let j_1 be the largest index such that $w_1/w_{j_1}\le\tau$ where $\tau\ge 1$ is a chosen threshold and we define $R_1:=\{j\in[n]\mid 1\le j\le j_1\}$. We then build R_2 similarly considering the indices $[n]\setminus R_1$ and proceed iteratively in this way. Then, fixed any $i\in[N]$ we build the family $\mathcal{S}_i$ by taking a prefix of the partition as H and the suffix of the partition as T. Formally, fixed $i\in[N]$, we consider the family:

$$\mathcal{S}_i:=\left\{\left(\bigcup_{t=i}^{i+j}R_t,\bigcup_{t=i+j+1}^{N}R_t\right)\,\middle|\,0\le j\le N-i\right\}.$$

Experimentally, we set $\tau = 3/2$. Note that it could be $|\mathcal{S}_i| = \Theta(N)$. To avoid a quadratic cost, we stop the inner For loop whenever we see that $\varepsilon_1 < 0.001$, where ε_1 is as defined in Line 8. This implies that the inner For loop executes at most $O(\log n)$ cycles (alternatively, one could consider $0 \le j \le \Theta(\log n)$ in the definition of $\mathcal{S}_i$). Indeed, by construction of $(R_1, \ldots, R_N)$, we have that for any i:

$$\min_{v \in R_i} w_v \ge \tau \cdot \max_{v \in R_{i+2}} w_v \quad \text{and} \quad \max_{v \in R_i} w_v \ge \tau \cdot \max_{v \in R_{i+1}} w_v.$$

Therefore, for $j = \lceil \log_\tau(\frac{n}{0.001}) \rceil = \Theta(\log n)$, we have:

$$\min_{v \in R_i} w_v \ge \tau^j \cdot \max_{v \in R_{i+j+1} \cup \cdots \cup R_N} w_v \ge \frac{n}{0.001} \cdot \max_{v \in R_{i+j+1} \cup \cdots \cup R_N} w_v.$$

Therefore, at the iteration $j = \lceil \log_\tau(\frac{n}{0.001}) \rceil$:

$$\varepsilon_1 := \frac{\sum_{v \in R_{i+j+1} \cup \cdots \cup R_N} w_v}{\min_{v \in R_i} w_v} \le \sum_{v \in R_{i+j+1} \cup \cdots \cup R_N} \frac{0.001}{n} \le 0.001,$$

concluding that we never make more than $O(\log n)$ iterations for any i.

Putting things together, for this particular way of choosing the partition and $\mathcal{S}$, Algorithm 5 can be implemented in $O(n \log n)$ time. Indeed, we can compute $\varepsilon_1, \varepsilon_2, \mu$ via range minimum/maximum queries (RMQ) on static arrays. It is well known that RMQ can be solved with $O(n \log n)$ preprocessing time and $O(1)$ query time via sparse tables (Bender and Farach-Colton, 2000). After the preprocessing phase, the remainder of the algorithm runs in time $O(n \log n)$ as we mentioned above.

G Omitted Proofs from Section 4

In this section, we provide our non-adaptive algorithm that learns almost all the pairs with $O(n \log^6 n)$ queries, thereby proving Theorem 4.3.

Recall that we denote with $\mathcal{G}(n,p)$ the Erdős–Rényi distribution over graphs with vertex set $[n]$ and edge probability $p \in [0,1]$, and that a random DAG from $\mathsf{DAG}(n,p)$ is obtained by first sampling a graph from $\mathcal{G}(n,p)$ and then orienting each edge $\{i,j\}$ from i to j if $i < j$. The number of comparable pairs in a DAG is the number of pairs of vertices $i < j$ such that there exists a path from i to j.

Recall that Pittel and Tungol (2001) showed that, in expectation, a large fraction of pairs are comparable (Lemma 4.1). By a simple application of Markov's inequality, this result can be boosted to hold with high probability.

Lemma G.1. *Choose any constant $A > 4$. Sample $G_1, \ldots, G_T \sim \mathsf{DAG}(n,p)$ independently, where $T = \lceil 2 \cdot \log_2(n) \rceil$ and $p = A \cdot \ln(n)/n$. Let $F_1, \ldots, F_T$ be the number of comparable pairs in $G_1, \ldots, G_T$, respectively. It holds that:*

$$\Pr\left[\max_{i \in [T]} F_i \le (1 - o(1)) \frac{n^2}{2} \left(1 - \frac{4}{A} + \frac{2}{A^2}\right)\right] \le \frac{1}{n^2}.$$

Proof. Consider $G \sim \mathsf{DAG}(n,p)$, and let $I = \binom{n}{2} - F$ where F is the number of comparable pairs in G. Then, by Lemma 4.1:

$$\mathbb{E}[I] = (1 \pm o(1)) \frac{n^2}{2} \left(\frac{2}{A} - \frac{1}{A^2}\right).$$

Therefore, we have:

$$\Pr\left[F \le (1 \pm o(1)) \frac{n^2}{2} \left(1 - \frac{4}{A} + \frac{2}{A^2}\right)\right] = \Pr\left[F \le \binom{n}{2} - 2 \cdot \mathbb{E}[I]\right] = \Pr\left[I \ge 2 \cdot \mathbb{E}[I]\right] \le \frac{1}{2},$$

where the last inequality follows by Markov's inequality. By independence:

$$\begin{aligned}\Pr\left[\max_{i \in [T]} F_i \le (1 - o(1)) \frac{n^2}{2} \left(1 - \frac{4}{A} + \frac{2}{A^2}\right)\right] &= \prod_{i=1}^{T} \Pr\left[F_i \le (1 - o(1)) \frac{n^2}{2} \left(1 - \frac{4}{A} + \frac{2}{A^2}\right)\right] \\ &\le 2^{-T} \le n^{-2}.\end{aligned}$$

□

This result tells us that for a large fraction of the pairs $\{i, j\}$, there exists in G a directed path from one vertex of $\{i, j\}$ to the other. In our analysis, we will consider a total order induced by sorting the items of the hidden MNL in increasing order of weight.

We also note the following standard result.

Lemma G.2. *Let $A > 1$ and $n \geq 3$, consider a graph G sampled from $\mathcal{G}(n, p)$, with $p = A \cdot \ln(n)/n$, let m be the number of edges in G (note that m is a random variable) then:*

$$\Pr\left[m \geq 2 \cdot A \cdot n \cdot \ln(n)\right] \leq n^{-\frac{A \cdot n}{9}}.$$

Proof. The standard Chernoff bound gives us that, if X is the sum of random variables in $\{0, 1\}$ with $\mathbb{E}[X] = \mu$ then:

$$\Pr[X \geq 2\mu] \leq e^{-\frac{\mu}{3}} \tag{10}$$

We note that m is binomially distributed with parameters $\binom{n}{2}$ and p. We then apply the bound to m to obtain:

$$\Pr\left[m \geq 2 \cdot A \cdot n \cdot \ln(n)\right] \leq \Pr\left[m \geq 2 \cdot \binom{n}{2} A \frac{\ln(n)}{n}\right] \overset{(10)}{\leq} \exp\left(-\frac{1}{3}\binom{n}{2} A \frac{\ln(n)}{n}\right) \leq e^{-\frac{An\ln(n)}{9}},$$

where the last inequality holds for $n \geq 3$. □

We now show that, with probability that tends to 1 as $n \to \infty$, the graph will not contain any really long path, which immediately implies that given most choices of $i < j$ there is a short path in G from i to j.

Lemma 4.2. *Let $G \sim \mathsf{DAG}(n, p)$ with $p = A \cdot \ln(n)/n$ and $A \geq 1$. Then, the probability that there exists no directed path in G of length at least $e^2 \cdot A \cdot \ln n$ is at least $1 - O\left(\frac{1}{n}\right)$.*

Proof. Let $B = e^2 \cdot A$. By the union bound, the probability that G contains a path of length greater than or equal to $B \ln n$ is at most:

$$\begin{aligned}
\sum_{\substack{P\text{: path} \\ \text{of length} \\ \geq \lceil B \ln n \rceil}} p^{|P|} &= \sum_{i=\lceil B \ln n \rceil}^{n-1} \binom{n}{i+1} \cdot p^i \leq \sum_{i=\lceil B \ln n \rceil}^{n-1} \left(\frac{e \cdot n}{i+1}\right)^{i+1} \cdot p^i \\
&= \sum_{i=\lceil B \ln n \rceil}^{n-1} \left(\frac{e \cdot n}{i+1}\right)^{i+1} \cdot \left(A \cdot \frac{\ln n}{n}\right)^i \\
&\leq \frac{e \cdot n}{B \ln n} \cdot \sum_{i=\lceil B \ln n \rceil}^{n-1} \left(\frac{A \cdot e \cdot \ln n}{i+1}\right)^i \\
&\leq \frac{e \cdot n}{B \ln n} \cdot \sum_{i=\lceil B \ln n \rceil}^{n-1} \left(\frac{A \cdot e}{B}\right)^i \\
&\leq \frac{e \cdot n}{B \ln n} \cdot \left(\frac{A \cdot e}{B}\right)^{B \ln n} \cdot \sum_{i=0}^{\infty} \left(\frac{A \cdot e}{B}\right)^i \\
&\leq \frac{e \cdot n}{B \ln n} \cdot \left(\frac{A \cdot e}{B}\right)^{B \ln n} \cdot \frac{B}{B - Ae} \\
&\leq \frac{e \cdot n}{B \ln n} \cdot e^{-2 \ln n} \cdot \frac{B}{B - Ae} \\
&\leq \frac{e}{B \cdot n \cdot \ln n} \cdot \frac{B}{B - Ae} \\
&= \frac{e^2}{(e-1) \cdot B \cdot n \cdot \ln n} = O\left(\frac{1}{n}\right),
\end{aligned}$$

where the first inequality uses that $\binom{n}{k} \leq \left(\frac{e \cdot n}{k}\right)^k$ for $1 \leq k \leq n$ and, the second-to-last inequality holds as long as $\frac{Ae}{B} \leq e^{-\frac{2}{B}}$ (and in particular, it holds for $B \geq A \cdot e^2$). □

Algorithm 6 Avg-Pairs-Preprocess(M, ε, δ)

Input: Sample access to an MNL M on $[n]$, an approximation parameter $\varepsilon \in (0, 1/3)$, and a constant value $\delta \in (0, 1/2)$.
Output: A data structure that can be used to approximate the winning distribution of a fraction $1 - \delta - o(1)$ of pairs
$A = \lceil 6/\delta \rceil$
$L = A \cdot e^2 \cdot \ln(n) + 1$
$\varepsilon_1 = \frac{\varepsilon}{6 \cdot L}$
$T = \lceil 2 \cdot \log_2(n) \rceil$
$p = A \cdot \ln(n)/n$
$p' = 1 - (1-p)^T$
$Q = Q(\varepsilon_1, \varepsilon_1, 1/n^3)$ as in Lemma G.3
Sample a graph $G \sim \mathcal{G}(n, p')$
Query Q times the MNL M on each pair $\{i, j\} \in E(G)$
for each pair $\{i, j\} \in E(G)$ **do**
    Let r be the output of the algorithm given by Lemma G.3 upon observing the results for the queries on $\{i, j\}$
    $r(i, j) = r$
    $r(j, i) = 1/r$ (if $r = 0$ then $r(j, i) = \infty$, and if $r = \infty$ then $r(j, i) = 0$)
end for
return G, $\{r(i, j), r(j, i)\}_{\{i,j\} \in E(G)}$, L, ε_1

Algorithm 7 Avg-Pairs-Predict(a, b)

Input: The output G, $\{r(i, j), r(j, i)\}_{\{i,j\} \in E(G)}$, L, and ε_1 of Algorithm 6, and a query $\{a, b\} \in \binom{[n]}{2}$.
Output: An estimate of $M_{\{a,b\}}(a)$ and $M_{\{a,b\}}(b)$, or an error
Using BFS, find a path $P = v_1, \ldots, v_\ell$ in G either from a to b, or from b to a having the following properties:

$$\ell \leq L \quad \text{and,} \quad \text{for all } k \in [\ell - 1]: \quad r(v_k, v_{k+1}) \leq 1 + \varepsilon_1 \tag{11}$$

if no such path P exists **then**
    return error
end if
$W = \prod_{k=1}^{\ell-1} r(v_k, v_{k+1})$
if P goes from a to b **then**
    return $\hat{M}_{\{a,b\}}(a) = \frac{W}{1+W}$, $\hat{M}_{\{a,b\}}(b) = \frac{1}{1+W}$
end if
return $\hat{M}_{\{a,b\}}(a) = \frac{1}{1+W}$, $\hat{M}_{\{a,b\}}(b) = \frac{W}{1+W}$

We will also make use of the following sub-routine, which is a simple application of the Chernoff bound:

Lemma G.3 (Lemma 13 from Chierichetti et al. (2026), paraphrased)**.** *Let M be an MNL supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$. Chosen any $i, j \in [n]$, $\alpha \in (0, 1)$, $\varepsilon \in (0, 1)$, and $\psi \in (0, 1/2)$, there exists a non-adaptive algorithm that makes $Q(\alpha, \varepsilon, \psi) = O(\frac{1}{\alpha\varepsilon^2} \log(1/\psi))$ queries to $\{i, j\}$ and that with probability $\geq 1 - \psi$, returns, upon observing the result of the queries, a value $r \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ such that:*

1. *If $\frac{w_i}{w_j} \leq \frac{1}{\alpha}$, then $r \neq \infty$, and if $\frac{w_i}{w_j} \geq \alpha$ then $r \neq 0$.*
2. *If $r \notin \{0, \infty\}$, $(1 - \varepsilon)\frac{w_i}{w_j} \leq r \leq (1 + \varepsilon)\frac{w_i}{w_j}$ and $(1 - \varepsilon)\frac{w_j}{w_i} \leq \frac{1}{r} \leq (1 + \varepsilon)\frac{w_j}{w_i}$.*

Our final algorithm has a preprocessing phase (Algorithm 6), and a prediction subroutine where it predicts the winning distribution for most of the pairs (Algorithm 7). Note that there can be some pairs for which our algorithm cannot provide a good estimate. We are finally ready to prove the main theorem of this section.

Theorem 4.3. *Choose any $\varepsilon \in (0,1)$, and any constant $\delta \in (0,1)$, and let M be an MNL over $[n]$ for a large enough n. There exists a non-adaptive algorithm that, with probability $1-o(1)$, makes at most $O\left(\frac{n\log^6 n}{\varepsilon^3}\right)$ queries to M on pairs and returns, for at least $(1-\delta-o(1))\binom{n}{2}$ pairs $\{a,b\}$, an estimate $\hat{M}_{\{a,b\}}(a)$ of $M_{\{a,b\}}(a)$ satisfying $|\hat{M}_{\{a,b\}}(a) - M_{\{a,b\}}(a)| \le \varepsilon$.*

Proof. Let us define $G_1, \ldots, G_T \sim \mathcal{G}(n,p)$. Note that $E(G)$, where $G \sim \mathcal{G}(n,p')$, has the same distribution as $\mathcal{E} = E(G_1) \cup \cdots \cup E(G_T)$. Indeed, each edge is in $\mathcal{E}$ independently with probability $1-(1-p)^T$.

We start by analyzing the query complexity of the algorithm. By Lemma G.2, with probability $\ge 1 - T \cdot n^{-\frac{A\cdot n}{9}} = 1-o(1)$, it holds that $|E(G_i)| \le 2 \cdot A \cdot n\ln(n)$ for each $i \in [T]$. Therefore, with the same probability:

$$|E(G)| \le \sum_{i=1}^{T} |E(G_i)| \le 2 \cdot T \cdot A \cdot n\ln(n) = O(n\log^2 n).$$

Then, with probability $1-o(1)$, the query complexity is:

$$|E(G)| \cdot Q(\varepsilon_1, \varepsilon_1, 1/n^3) = O(n\log^6(n)\varepsilon^{-3}).$$

We now prove that the algorithm returns an ε-additive approximation for most pairs. Without loss of generality, we rename the items of the MNL so that $w_1 \le w_2 \le \cdots \le w_n$. By Lemma G.1, with probability $\ge 1 - 1/n^2$, there exists at least one $G^* \in \{G_1, \ldots, G_T\}$ such that:

$$\begin{aligned} F^* &\ge (1-o(1))\frac{n^2}{2}\left(1 - \frac{4}{A} + \frac{2}{A^2}\right) \\ &\ge (1-o(1))(1-\delta)\frac{n^2}{2} \\ &= (1-\delta-o(1)) \cdot \binom{n}{2}, \end{aligned} \tag{12}$$

where F^* is the number of comparable pairs in the DAG induced by G^*. Moreover, by Lemma 4.2, with probability at least $1 - O(T/n) = 1-o(1)$, every path in all DAGs induced by $\{G_1, \ldots, G_T\}$ has length at most $A \cdot e^2 \cdot \ln(n)$ (which is at most $L-1$). Consider any comparable pair $\{a,b\}$ in G^*, assuming that $a < b$. By the previous observation and by definition of comparable pair, there exists a path $P_{\{a,b\}} = u_1, \ldots, u_\ell$ in G^* with $u_1 = a$, $u_\ell = b$, such that:

$$\ell \le L \quad \text{and for all } k \in [\ell-1]: \quad w_{u_k} \le w_{u_{k+1}}, \tag{13}$$

where the second property holds since $u_i \le u_{i+1}$ by definition of comparable pair. Observe also that, since G has at most n^2 edges, all the invocations of Lemma G.3 at Line 13 are correct with probability at least $1 - n^2/n^3 = 1 - 1/n$ by our choice of Q. We condition on this event for the remainder of the proof. We now show that any path with the properties of Equation (13) also has the properties of Equation (11). Indeed, if $w_{u_k} \le w_{u_{k+1}}$, by Lemma G.3, it must hold that either:

$$r(u_k, u_{k+1}) = 0, \quad \text{or} \quad r(u_k, u_{k+1}) \le \frac{w_{u_k}}{w_{u_{k+1}}}(1+\varepsilon_1) \le 1+\varepsilon_1.$$

Therefore, any path with properties (13) also has properties (11). Moreover, by Equation (12), there are $(1-\delta-o(1))\binom{n}{2}$ pairs with a path having properties (13), and therefore there at least that many pairs with a path having properties (11).

To conclude the proof, we now show that whenever the algorithm finds a path $P = v_1, \ldots, v_\ell$ as in Equation (11), it correctly returns an ε-additive estimate of $M_{\{a,b\}}(a)$. We consider the case where P goes from a to b, the other case is completely symmetrical. For simplicity, let us relabel the items so that $v_i = i$ for each $i \in [\ell]$, and in particular $a = 1$ and $b = \ell$. Consider first the case where $W = 0$. Then, there exists $k^* \in [\ell-1]$ such that $r(k^*, k^*+1) = 0$, which by Lemma G.3 implies that $w_{k^*}/w_{k^*+1} \le \varepsilon_1$. Moreover, by Equation (11), we have that for each $j \in [\ell-1]$:

$$w_j \le \left(\frac{1+\varepsilon_1}{1-\varepsilon_1}\right) w_{j+1} \le (1+3\varepsilon_1)w_{j+1},$$

where we use that $\varepsilon_1 \leq 1/3$ and that either $r(j, j+1) = 0$ or $r(j, j+1) \geq \frac{w_j}{w_{j+1}}(1-\varepsilon_1)$. We then have that:

$$\begin{aligned}\frac{w_a}{w_b} = \frac{w_1}{w_\ell} = \prod_{i=1}^{\ell-1} \frac{w_i}{w_{i+1}} &\leq (1+3\varepsilon_1)^{\ell-2} \cdot \varepsilon_1 \\ &\leq (1+6(\ell-2)\varepsilon_1) \cdot \varepsilon_1 \leq 2\varepsilon_1 \leq \varepsilon,\end{aligned}$$

where we used that $\ell \leq L$, $\varepsilon_1 \leq \frac{\varepsilon}{6 \cdot L}$, and $(1+\alpha)^\beta \leq 1 + 2\alpha\beta$ for $\beta > 0$, $\alpha \in (0,1)$, $2\alpha\beta \in (0,1)$. This in turn implies:

$$M_{\{a,b\}}(a) \leq \varepsilon,$$

therefore, $\hat{M}_{\{a,b\}}(a) = 0$ is an ε-additive estimate for $M_{\{a,b\}}(a)$.

We now turn to the case $W > 0$. By Equation (11) and Lemma G.3 we have that for each $j \in [\ell - 1]$, $\frac{w_j}{w_{j+1}}(1-\varepsilon_1) \leq r(j, j+1) \leq \frac{w_j}{w_{j+1}}(1+\varepsilon_1)$. Therefore:

$$W = \prod_{i=1}^{\ell-1} r(i, i+1) \leq (1+\varepsilon_1)^{\ell-1} \cdot \frac{w_1}{w_\ell} \leq (1+2(\ell-1)\varepsilon_1) \cdot \frac{w_a}{w_b} \leq (1+2L \cdot \varepsilon_1) \cdot \frac{w_a}{w_b},$$

and:

$$W \geq (1-\varepsilon_1)^{\ell-1} \cdot \frac{w_1}{w_\ell} \geq (1-(\ell-1)\varepsilon_1)\frac{w_a}{w_b} \geq (1 - L \cdot \varepsilon_1) \cdot \frac{w_a}{w_b},$$

where we used Bernoulli's inequality: $(1-x)^t \geq 1 - t \cdot x$ for $x < 1$. Therefore, we have:

$$\begin{aligned}\hat{M}_{\{a,b\}}(b) = \frac{1}{1+W} &\leq \frac{1}{1+\frac{w_a}{w_b}(1-L\varepsilon_1)} \\ &\leq \frac{1}{1+\frac{w_a}{w_b}}(1+2L\varepsilon_1) \\ &= M_{\{a,b\}}(b) + 2L \cdot M_{\{a,b\}}(b) \cdot \varepsilon_1 \\ &\leq M_{\{a,b\}}(b) + \varepsilon,\end{aligned}$$

where we used that $\frac{1}{1+x(1-y)} \leq \frac{1}{1+x} \cdot (1+2y)$ for $x \geq 0, y \in (0, 1/2)$. Similarly,

$$\begin{aligned}\hat{M}_{\{a,b\}}(b) = \frac{1}{1+W} &\geq \frac{1}{1+\frac{w_a}{w_b}(1+2L\varepsilon_1)} \\ &\geq \frac{1}{1+\frac{w_a}{w_b}} \cdot (1-2L\varepsilon_1) \\ &= M_{\{a,b\}}(b) \cdot (1-2L\varepsilon_1) \\ &\geq M_{\{a,b\}}(b) - \varepsilon,\end{aligned}$$

where we used $\frac{1}{1+x(1+y)} \geq \frac{1}{1+x} \cdot (1-y)$ for $x > 0, y \in (0,1)$. By symmetry we have that:

$$\left| M_{\{a,b\}}(a) - \hat{M}_{\{a,b\}}(a) \right| \leq \varepsilon.$$

Therefore, the algorithm returns an ε-additive estimate whenever it finds a path with the properties of Equation (11). □

H Missing Constructions

In this section, we explicitly show some simple constructions that were claimed in the main body without a formal proof.

Full slate error is not representative of worst-case error. We show that two MNLs can be close on the full slate but very different over pairs.

Observation H.1. For each $p \geq 1$, there exists MNLs M, M', and M'' such that $\|M_{[n]} - M'_{[n]}\|_p \leq \|M_{[n]} - M''_{[n]}\|_p$ but $d_\infty(M, M') > d_\infty(M, M'') + c$ for some constant $c > 0$.

Proof. Fix an arbitrarily small $\delta > 0$ and set $\varepsilon = \delta/n$. Define MNL M with weights $1, \varepsilon, \ldots, \varepsilon$ (where ε is repeated $n-1$ times), M' with weights $1, 2\varepsilon, \ldots, 2\varepsilon, 4\varepsilon$, and M'' with weights $1, 4\varepsilon, \ldots, 4\varepsilon$. It is clear that the full slate error is smaller between M and M' than between M and M'' (under any ℓ_p norm). However, for the slate containing only the last two items, the winning distribution of M is $(1/2, 1/2)$, while in M' is $(1/3, 2/3)$. Thus, $d_\infty(M, M') \geq 1/6$. Instead, for each slate the error between M and M'' is always bounded by $O(\delta)$ (even in ℓ_1) and therefore it can be made arbitrarily small. □

The worst error on pairs is not always the worst error overall. We observed experimentally that the worst error over pairs is a good indicator of the worst error over all slates. In this section, we show that there exists some MNLs for which this is not true. Still, it is interesting that in real-world instances such boundary cases does not seem to occur.

Observation H.2. There exists MNLs M and M' such that $d_\infty^{\text{pair}}(M, M') < d_\infty(M, M') - c$ for some constant $c > 0$.

Proof. consider MNL M with weights $1, \varepsilon, \ldots, \varepsilon$ and MNL M' with weights $2, \varepsilon, \ldots, \varepsilon$, where $\varepsilon = 1/(n-1)$ and it is repeated $n-1$ times. Then, $d_\infty^{\text{pair}}(M, M') = O(1/n)$, but the error on the full slate (and other large slates) is $\Omega(1)$. □

Non-Adaptive algorithms that make $o(n^2)$ pair queries must err on $\Omega(n)$ pairs. We briefly show that any non-adaptive algorithm that makes $o(n^2)$ queries must, in the worst-case over the choice of MNL instance, produce a poor estimate on $\Omega(n)$ pairs of elements.

We prove the following result.

Theorem H.3. *Fix any sufficiently large even number n. Let $\mathcal{A}$ be a non-adaptive algorithm that queries fewer than $\frac{1}{10}\binom{n}{2}$ pairs from a hidden MNL M supported on $[n]$ and then returns a learned MNL $\hat{M}$. Then, there exists an MNL M such that*

$$\mathbb{E}\left[\left|\left\{\{i,j\} \in \binom{[n]}{2} \mid \|M_{\{i,j\}} - \hat{M}_{\{i,j\}}\|_\infty \geq \frac{1}{3}\right\}\right|\right] \geq \Omega(n).$$

Proof. By Yao's minimax principle, we may assume that the set of queries made by the algorithm is fixed, and we design a probability distribution over MNLs with the desired property. Let $W = 10n^2$.

Consider an MNL constructed stochastically as follows: one first samples a random perfect matching $\mathcal{R}$ (a partition into pairs) of the elements of $[n]$. Let these pairs be $p_1 = \{f_1, s_1\}, \ldots, p_{n/2} = \{f_{n/2}, s_{n/2}\}$, then, for each pair p_i we sample a Bernoulli random variable $X_i \sim \text{Bernoulli}(1/2)$. We then assign weights:

$$w_{f_i} = \frac{99}{100}W^{2i}, \qquad w_{s_i} = \frac{1}{100}W^{2i}$$

if $X_i = 1$, or:

$$w_{f_i} = \frac{1}{100}W^{2i}, \qquad w_{s_i} = \frac{99}{100}W^{2i}$$

if $X_i = 0$.

Let Y be the number of pairs among $\{p_1, \ldots, p_{n/2}\}$ that are queried by the algorithm. By linearity of expectation, we have $\mathbb{E}[Y] \leq \frac{n}{2} \cdot \frac{1}{10} = \frac{n}{20}$.

Let $\mathcal{E}$ be the event that the winner in every query made by the algorithm is part of the matching pair with the highest index (e.g. if the algorithm queries the pair f_1, s_5 then the winner is s_5). For each pair, this happens with probability at least $\frac{1}{1+99/W^2}$. And hence by the union bound $\Pr[\mathcal{E}] \geq 1 - \frac{99}{W^2} \cdot \binom{n}{2}$. Conditioned on $\mathcal{E}$, the algorithm may be able to infer which pair an item belongs to in the matching, but has no information about the value of X_i unless the exact pair $\{f_i, s_i\}$ is queried, and hence, for each pair p_i not queried, its chance of producing an estimate $\hat{M}_{p_i}$ such that $\|\hat{M}_{p_i} - M_{p_i}\|_\infty \leq \frac{1}{3}$ is at most $\frac{1}{2}$.

Let:

$$P = \left|\left\{i \in [n/2] \mid \|M_{p_i} - \hat{M}_{p_i}\|_\infty \leq \frac{1}{3}\right\}\right|$$

Then:

$$\begin{aligned}\mathbb{E}[P] &= \mathbb{E}[P \mid \mathcal{E}]\Pr[\mathcal{E}] + \mathbb{E}\left[P \mid \overline{\mathcal{E}}\right]\Pr[\overline{\mathcal{E}}] \\ &\leq \frac{n}{20} + \frac{1}{2}\cdot\frac{n}{2} + \frac{99}{W^2}\cdot\binom{n}{2}\cdot\frac{n}{2} \\ &\leq \frac{7}{20}\cdot n,\end{aligned}$$

and the result follows. □

I A Practical Heuristic for Exact Maximum Distance on Pairs

In this section, we show an algorithm which exactly computes $d_\infty^{\text{pair}}(M, M')$, but may take $O(n^2 \log n)$ time in the worst case. In practice, we found its pruning strategy to yield very fast running times on some instances arising from real-world datasets.

This heuristic algorithm was discovered with the assistance of ChatGPT 5, and later written and verified by the authors. To the best of our knowledge this was not known in the literature.

Theorem I.1. *Algorithm 8 correctly finds the value of $d_\infty^{\text{pair}}(M, M')$ as well as a pair $\{i, j\} \subseteq [n]$ such that:*

$$d_\infty^{\text{pair}}(M, M') = \|M_{\{i,j\}} - M'_{\{i,j\}}\|_\infty,$$

and has a worst-case running time of $O(n^2 \log n)$.

We begin by proving a technical lemma. Recall the definition of the logistic function:

$$\sigma(t) := \frac{1}{1 + e^{-t}},$$

and the hyperbolic tangent:

$$\tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}.$$

Lemma I.2. *For any $t, d \in \mathbb{R}$ we have:*

$$|\sigma(t + d) - \sigma(t)| \leq \tanh\left(\frac{|d|}{4}\right)$$

Proof. Recall the definitions of cosh and sinh:

$$\cosh(t) := \frac{e^t + e^{-t}}{2} \qquad\qquad \sinh(t) := \frac{e^t - e^{-t}}{2}$$

as well as the following identities that follow from the definitions:

$$\sigma(t) = \frac{1 + \tanh(t/2)}{2} \tag{14}$$

$$\tanh(t_1) - \tanh(t_2) = \frac{\sinh(t_1 - t_2)}{\cosh t_1 \cosh t_2} \tag{15}$$

$$\cosh(t_1)\cosh(t_2) = \frac{1}{2}\left(\cosh(t_1 + t_2) + \cosh(t_1 - t_2)\right) \tag{16}$$

$$\tanh\left(\frac{t}{2}\right) = \frac{\sinh t}{1 + \cosh(t)} \tag{17}$$

Fix any $t, d \in \mathbb{R}$:

$$|\sigma(t + d) - \sigma(t)| \overset{(14)}{=} \frac{1}{2}\left|\tanh\left(\frac{t + d}{2}\right) - \tanh\left(\frac{t}{2}\right)\right|$$

Algorithm 8 `ExactMaxPairDistance`(M, M')

Input: The weights $\{w_i\}_{i\in[n]}$ of a MNL M and the weights $\{v_i\}_{i\in[n]}$ of a MNL M'.
Output: The value of $d_\infty^{\text{pair}}(M, M')$, as well as a pair $P = \{i, j\} \subseteq [n]$ such that $\|M_P - M'_P\|_\infty = d_\infty^{\text{pair}}(M, M')$.
For each $i \in [n]$ compute $c_i = \ln w_i - \ln v_i$,
Let K be an array containing the numbers i in $[n]$ sorted in increasing order of c_i.
Let Q be an empty Priority Queue, place the pair $(K[1], K[n])$ in Q with priority $\tanh\left(\frac{|c_{K[n]} - c_{K[1]}|}{4}\right)$
Let S be a dictionary, add $(K[1], K[n])$ to S.
$max_dist = 0$, $max_pair = \{K[1], K[n]\}$
while Q is not empty **do**
  Pop the pair $(K[i], K[j])$ with the largest key value k_{ij} from Q
  if $k_{ij} < max_dist$ **then**
    continue from the next iteration
  end if
  $dist = \|M_{\{K[i],K[j]\}} - M'_{\{K[i],K[j]\}}\|_\infty$
  if $dist > max_dist$ **then**
    $max_dist = dist$
    $max_pair = \{K[i], K[j]\}$
  end if
  if $i = j - 1$ **then**
    continue
  end if
  if $(K[i + 1], K[j]) \notin S$ **then**
    Add $(K[i + 1], K[j])$ to S
    $k_{i+1,j} = \tanh\left(\frac{|c_{K[j]} - c_{K[i+1]}|}{4}\right)$
    if $k_{i+1,j} > max_dist$ **then**
      Add $(K[i + 1], K[j])$ to Q with key $k_{i+1,j}$
    end if
  end if
  if $(K[i], K[j - 1]) \notin S$ **then**
    Add $(K[i], K[j - 1])$ to S
    $k_{i,j-1} = \tanh\left(\frac{|c_{K[j-1]} - c_{K[i]}|}{4}\right)$
    if $k_{i,j-1} > max_dist$ **then**
      Add $(K[i], K[j - 1])$ to Q with key $k_{i,j-1}$
    end if
  end if
end while
return (max_dist, max_pair)

$$\overset{(15)}{=} \frac{|\sinh\left(\frac{d}{2}\right)|}{2\cosh\left(\frac{t+d}{2}\right)\cosh\left(\frac{t}{2}\right)}$$
$$\overset{(16)}{=} \frac{|\sinh\left(\frac{d}{2}\right)|}{\cosh\left(t + \frac{d}{2}\right) + \cosh\left(\frac{d}{2}\right)}.$$

Hence:

$$\max_{t\in\mathbb{R}} |\sigma(t + d) - \sigma(t)| = \frac{\sinh(|d|/2)}{1 + \cosh(|d|/2)} \overset{(17)}{=} \tanh\left(\frac{|d|}{4}\right).$$

And, in particular:

$$\forall t \in \mathbb{R} : |\sigma(t + d) - \sigma(t)| \leq \tanh\left(\frac{|d|}{4}\right). \qquad \square$$

Proof of Theorem I.1. First, note that the inner loop does at most $O(\log n)$ operations for each pair that gets popped from Q (due to the $O(\log n)$ complexity of popping and adding to a priority queue).

But each pair is only added to Q once, and hence the loop is executed at most $\binom{n}{2}$ times, and the result follows.

The algorithm progressively inserts all the pairs into Q, then extracts each pair P and computes the value of $\|M_P - M'_P\|_\infty$ to check if it is higher than the highest value found so far, if so it saves the result. The algorithm starts off with the pairs $\{K[1], K[n]\}$ and upon extracting the pair $\{K[i], K[j]\}$ from the queue, it attempts to add the pairs $\{K[i+1], K[j]\}$ and $\{K[i], K[j-1]\}$ to the queue. This way, every pair is eventually added to Q, with a single exception.

If, upon extracting $\{K[i], K[j]\}$ the algorithm finds that its key $\tanh\left(\frac{|c_{K[j]} - c_{K[i]}|}{4}\right)$ is smaller than max_dist, then the algorithm will not attempt to add $\{K[i+1], K[j]\}$ and $\{K[i], K[j-1]\}$ to the queue. This is because, as we now prove, $\tanh\left(\frac{|c_{K[j]} - c_{K[i]}|}{4}\right)$ is monotone function of $|c_{K[j]} - c_{K[i]}|$, and an upper bound to $\|M_{\{K[i],K[j]\}} - M'_{\{K[i],K[j]\}}\|_\infty$, and hence, if $\tanh\left(\frac{|c_{K[j]} - c_{K[i]}|}{4}\right) \leq max_dist$, then, for all pairs i_1, j_i with $i \leq i_1 < j_1 \leq j$ we will have:

$$\|M_{\{K[i_1],K[j_1]\}} - M'_{\{K[i_1],K[j_1]\}}\|_\infty < max_dist,$$

and hence there is no need to check any of those pairs.

We are left with proving that $\tanh\left(\frac{|c_{K[j]} - c_{K[i]}|}{4}\right)$ is always an upper bound to $\|M_{\{K[i],K[j]\}} - M'_{\{K[i],K[j]\}}\|_\infty$. Let

$$a_i = \ln w_i, \qquad b_i = \ln v_i \qquad \text{and} \qquad c_i = a_i - b_i = \ln\left(\frac{w_i}{v_i}\right)$$

Moreover, for any pair of elements $i, j \in [n]$, define $x_{ij} = a_i - a_j$ and $y_{ij} = b_i - b_j$ and note that $x_{ij} - y_{ij} = c_i - c_j$. By Lemma I.2:

$$\begin{aligned}
\left|\frac{w_i}{w_i + w_j} - \frac{v_i}{v_i + v_j}\right| &= \left|\frac{1}{1 + \frac{w_j}{w_i}} - \frac{1}{1 + \frac{v_j}{v_i}}\right| \\
&= |\sigma(x_{ij}) - \sigma(y_{ij})| \\
&\leq \tanh\left(\frac{|x_{ij} - y_{ij}|}{4}\right) \\
&= \tanh\left(\frac{|c_i - c_j|}{4}\right),
\end{aligned}$$

and the theorem follows. □

J Missing Experiments

J.1 Hyperparameters and Implementation Details

For algorithms ILSR and MM, we use the Python implementation available in the Choix library[10] (Maystre and Grossglauser, 2015). We set the regularization parameter to $\alpha = 10^{-5}$ for both algorithms. We tested also $\alpha \in \{10^{-6}, 10^{-4}, 10^{-1}\}$ on the tennis datasets, but $\alpha \in \{10^{-4}, 10^{-6}\}$ performed comparable to 10^{-5} and $\alpha = 10^{-1}$ performed much worse. For the non-adaptive methods, we declare convergence when the relative ℓ_1-distance between the weights of two successive iterations is smaller than 10^{-8}.[11]. We set a maximum number of iterations of 100 for ILSR and 10^4 for MM (which is known to converge much slower). The methods occasionally hit the maximum number of iterations when running with the smallest query budget. For Lobster, we provide our Python implementation, since one was not available before.

Recall that we first run Lobster and then run the non-adaptive algorithms with the same query budget. We use parameters $\varepsilon \in \{0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 1\}$. To accelerate the convergence of MM on Codeforces, for $\varepsilon = 0.005$ we feed to MM the true hidden weights of the MNL as the initial

[10] `https://choix.lum.li/en/latest/`

[11] This is the standard check used in the Choix library

Table 1: Number of items in the datasets, and the ratio between the maximum and minimum weight of the items.

DATASET	n	$\frac{\max_{i\in[n]} w_i}{\min_{i\in[n]} w_i}$
TENNIS-WOMEN	508	636
TENNIS-WOMEN-2025	451	253
TENNIS-MEN	485	643
TENNIS-MEN-2025	456	551
CHESS-BULLET	11378	$154 \cdot 10^6$
CHESS-BLITZ	13446	$72 \cdot 10^6$
CODEFORCES-COMPETING	52932	$4.2 \cdot 10^9$
CODEFORCES-ACTIVE	148158	$4.8 \cdot 10^9$
CODEFORCES-FULL	842045	$4.9 \cdot 10^9$

weights to optimize. Instead, for all other ε's we feed as starting weights for MM the output weights obtained with the ε immediately smaller. Recall that the likelihood is convex, we adopt this trick only to make MM run faster on this large dataset. Note that providing the true weights in input for MM might give this algorithm an unfair advantage. Even with this advantage we observe Lobster to perform better than MM on Codeforces, justifying this initialization that was only required for computational reasons.

All experiments have been run either on a Macbook Air M3 with 16Gb of RAM, or on a desktop computer with an Intel Xeon(R) CPU E5-2650 and 64Gb of RAM.

J.2 Datasets Statistics

In Table 1, we report the number of items and the ratio $\max_{i\in[n]} w_i / \min_{i\in[n]} w_i$ in each dataset. Observe that the smallest datasets of tennis players also have a significantly smaller ratio. This is an additional factor explaining why ILSR performs better than Lobster on tennis datasets.

On the other hand, we also observe, in Appendix J.4, that ILSR outperforms Lobster when the Chess and Codeforces datasets are sparsified to fewer than 1000 items. Our sparsification procedure selects items at uniform intervals after sorting, and hence it approximately preserves the ratio between the maximum and minimum weights. Taken together, these results indicate that, in determining whether ILSR outperforms Lobster, both dataset size and weight ratios are crucial factors.

J.3 Missing Plots from Section 3.2

In this section, we report the full set of results for the experiment described in Section 3.2. Specifically, we provide Figure 3, Figure 4, Figure 5, Figure 6, and Figure 7. We present results for the metric d_1. The corresponding results for d_∞ are qualitatively similar and are omitted for brevity. The results are similar to those discussed in Section 3.2.

On top of the error induced by the worst pair, our lower and upper bound on d_1, the average error on the pairs, and the ℓ_1-error on the full slate $\|M_{[n]} - M'_{[n]}\|_1$, we also report two more errors previously considered in the literature. Specifically, if M is the ground truth MNL with weights $\{w_i\}_{i\in[n]}$ and M' is the learned MNL with weights $\{w'_i\}_{i\in[n]}$, we consider the root mean squared error (RMSE) used in the experiments of Maystre and Grossglauser (2015) to measure the quality of the estimated MNL:

$$\frac{1}{\sqrt{n}} \cdot \|\theta - \theta'\|_2,$$

where $\theta_i := \ln(w_i) - t$ and t is such that $\sum_{i\in[n]} \theta_i = 0$ (and, similarly, $\theta'_i = \ln(w'_i) - t'$). We also compute the following normalized ℓ_2 distance on the full slate which was used in (Negahban et al., 2012):

$$\frac{\|M_{[n]} - M'_{[n]}\|_2}{\|M_{[n]}\|_2}.$$

We observe that, similarly, to the ℓ_1 error on the full slate, these distances can be very small even though there are pairs of items incurring a large ℓ_1 error—meaning that these are not good proxies for the worst-case error.

J.4 Adaptive vs Non-Adaptive Algorithms with Sparsified Datasets

From the experiments in Section 3.2, we know that the adaptive algorithm Lobster performs better than the non-adaptive one ILSR on datasets with at least 10^4 items, while it performs worse on datasets of ≈ 500 items. The goal of this section is to test dataset sizes between 500 and 10^4 to better understand the regime in which adaptivity is preferable to non-adaptivity. We consider the five datasets with more than 10^4 items: `chess-blitz`, `chess-bullet`, `codeforces-competing`, `codeforces-active`, and `codeforces-full`. For each of them, we sparsify the dataset to have k items by first sorting the items by weight and then taking k evenly spaced items. We test k from 250 to 5000 with steps of 250 items. Similarly to Section 3.2, we first run Lobster (we set $\varepsilon = 0.1$) and then run ILSR with a query budget equal to the number of queries made by Lobster. We repeat each experiment 10 times, and report average results and standard deviations. We report the results in Figure 8.

We observe that from $n \approx 1000$ onward, the adaptive algorithm performs better than the non-adaptive one in all cases.

J.5 Uniform at Random Sampling vs Sampling with Repetitions

In our experiments on Chess and Tennis, the queries for the non-adaptive algorithms have been selected uniformly at random with replacement among the $\binom{n}{2}$ possible pairs. Due to memory costs, for Codeforces we resorted to a different sampling: we first sample at most $10 \cdot n \ln n$ pairs uniformly at random, and then distribute the remaining query budget approximately uniformly among the sampled pairs. This second sampling scheme is also what we use in Section 4 to prove that non-adaptive algorithms can approximate almost all pairs with $\tilde{O}(n)$ queries.

In this section, we compare the performances of these two sampling schemes on datasets Tennis and Chess. We report the results in Figure 9. We observe that the two sampling schemes behave roughly the same, with the uniform at random sampling being slightly preferable in some cases.

J.6 Error Distribution on Pairs

In the experiments of Section 3.2, we observed that the error of the worst pair is close to the worst possible error over slates of every cardinality. However, the *average* error on pairs is very small. A natural question is whether one should worry about these "bad" pairs or if these pairs are uninteresting to approximate. Clearly, this is application-dependent.

In competitions, where our datasets originate from, one is typically interested in predicting the winner between two similarly skilled players (accurately predicting the outcome of a match between a professional player and an amateur is certainly easier, and somewhat less impressive). One may therefore consider a pair of items interesting if the two items involved are of similar weight. In this set of experiments, we will narrow in on a collection $\mathcal{P}$ of pairs of items with this property: $\mathcal{P} := \{(i,j) \mid w_i \leq w_j \leq \alpha \cdot w_i\}$, where we set $\alpha = 1.01$ and $\{w_i\}_{i \in [n]}$ are the weights of the ground truth MNL. For efficiency reasons, for each i we consider at most 50 j's. We then compute the error distribution over the pairs in $\mathcal{P}$ and compare it with $|\mathcal{P}|$ pairs sampled uniformly at random.

Just like in Section 3.2, having fixed an MNL M, we learn another MNL M' and look at the distance between them. We set the number of queries to learn M' to $\approx 10^3 \cdot n \log_2 n$, corresponding to $\varepsilon = 0.1$ in the experiments of Section 3.2. We report the results in Figure 10, Figure 11, and Figure 12. We present results for the metric d_1. The corresponding results for d_∞ are qualitatively similar and are omitted for brevity.

We observe that the pairs in $\mathcal{P}$ incur significantly higher errors than random pairs. Moreover, even among items of comparable weights, there are some pairs that incur a very large error: roughly half the worst possible on pairs. However, we also observe that most pairs (both from $\mathcal{P}$ and the whole universe $[n]$) incur a very small error, again in accordance with Theorem 4.3 in Section 4.

We can therefore conclude that non-adaptive algorithms give a good approximation for most pairs, but fail to uniformly approximate every pair even among items of similar weight: if approximating the behavior of an MNL on all pairs is critical, adaptive algorithms will be a better choice at scale.

J.7 Comparison of Average Error for Adaptive and Non-Adaptive Algorithms

In Section 3.2 we have seen that the Lobster adaptive algorithm achieves a smaller worst-case error compared to non-adaptive methods. A natural question is whether this improvement in the worst-case comes at a cost on the average error. In this section, we compare the average pairwise error incurred by Lobster and that incurred by ILSR. We report the results in Table 2 for the `chess-bullet` dataset; the results for the other datasets are similar. We observe that while Lobster performs better on the worst pair, ILSR has a smaller average pairwise error. However, in absolute terms, the average pairwise error in very small for both algorithms, and therefore adaptivity is still preferable.

J.8 Experiments on Power-Law Synthetic Dataset

The real-world datasets we used stem from competitions. In this section, we perform experiments on heavy-tailed MNLs, with the goal of mimicking a product catalog. Specifically, we consider an MNL with n weights, where $w_i \propto i^{-\alpha}$ for $i \in [n]$ and a constant $\alpha \geq 0$. Note that larger values of α increase the ratio between the maximum and minimum item weight, while at $\alpha = 0$ the MNL is uniform.

Real vs Learned MNL. Similarly to Section 3.2, we analyze the error incurred by different algorithms in approximating the real MNL. We consider the case with $n = 5000$ and $\alpha = 1$ which corresponds to the standard zipfian case. We report in Figure 13 the results for both ILSR and Lobster. We observe that, similarly to the real-world datasets, the worst-pair error is a better proxy for the worst-case performance than the full-slate error. It is worth highlighting that for ILSR, the local search finds a slate significantly worse than the worst-pair; while instead for Lobster the worst-pair essentially coincides with the worst error found by the local search.

Disentangling Dataset Size and Max/Min Ratio. We have seen that the advantage of adaptive algorithms over non-adaptive ones is conditioned on the dataset size, n, and the max/min ratio of its weights. In our real-world datasets, large datasets also tended to have larger max/min ratio. Here, we try to disentangle the effect of these two components, which we can do since the dataset is synthetic. We do so by keeping n fixed and varying α, which controls the ratio. In particular, we tested $\alpha \in \{0.1, 0.5, 1, 2\}$ and $n \in \{250, 5000\}$. In order to let the max/min ratio depend only on α, and not on n, we always set $w_1 = 1^{-\alpha}$ and $w_n = 5000^{-\alpha}$ and interpolate for the other items with uniform steps. The max/min ratios corresponding to our choice of α are: 2.7, 70.7, 5000, and $25 \cdot 10^6$. We run Lobster with $\varepsilon = 0.1$ and then run ILSR with the same number of queries. We report the results in Figure 14.

We can observe that at $n = 250$, ILSR is always better than Lobster independently of the max/min ratio. This is a phenomenon already observed in Appendix J.4.

On the other hand, at $n = 5000$, ILSR is preferable with small values of α (i.e., when the dataset is closer to being uniform), while Lobster becomes significantly better as the max/min ratio increases.

In summary, for small values of n or max/min ratio, non-adaptivity is sufficient, while when the dataset is large in size and has a broad range of values, adaptivity becomes superior.

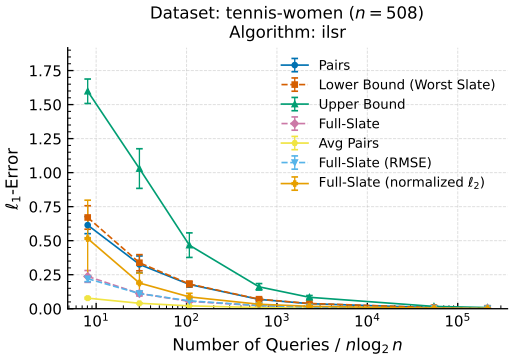
Exact d_1 for small scale datasets. Since the metrics d_1 and d_∞ cannot be computed exactly efficiently (unless $P = NP$) on our large datasets, we do not know whether the true d_1 error is closer to our computed upper bound or lower bound (when there is a significant gap between them). The best algorithm we know to compute d_1 exactly is to simply try all slates incurring into a running time of $O(n \cdot 2^n)$. We compute d_1 exactly on this synthetic power-law dataset for $n = 20$ and compare it to the bounds found by our efficient algorithms. Results for Lobster are reported in Table 3, those for ILSR are similar. We observe that, on this small dataset, the local search always finds the exact worst slate and in general the lower bound we compute is closer to the true worst error than the upper bound.

Table 2: Pairwise error comparison of ILSR and Lobster. We first run Lobster with a chosen ε parameter and then run ILSR with a query budget equal to the number of queries performed by Lobster.

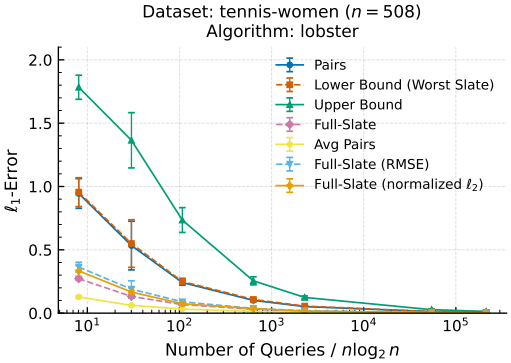
	$\varepsilon = 1$	$\varepsilon = 0.5$	$\varepsilon = 0.25$	$\varepsilon = 0.1$	$\varepsilon = 0.05$	$\varepsilon = 0.01$	$\varepsilon = 0.005$
ILSR (worst pair)	1.83	1.41	0.91	0.41	0.19	0.04	0.013
Lobster (worst pair)	1.11	0.72	0.31	0.12	0.06	0.01	0.006
ILSR (avg pair)	0.05	0.03	0.01	0.006	0.003	0.0006	0.0003
Lobster (avg pair)	0.09	0.04	0.02	0.009	0.005	0.0011	0.0005

Table 3: Comparison of the exact d_1 error computed by brute force, and the error estimated via our algorithms. The dataset is the synthetic power-law dataset with $n = 20$ and the algorithm used is Lobster.

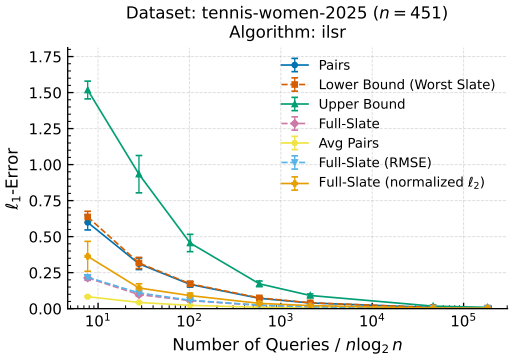
	$\varepsilon = 1$	$\varepsilon = 0.5$	$\varepsilon = 0.25$	$\varepsilon = 0.1$	$\varepsilon = 0.05$	$\varepsilon = 0.01$	$\varepsilon = 0.005$
worst pair	0.5339	0.2992	0.1304	0.0630	0.0376	0.0077	0.0032
local search	0.5396	0.3106	0.1348	0.0636	0.0390	0.0080	0.0035
upper bound	1.3803	0.8597	0.3364	0.1463	0.0832	0.0174	0.0082
exact d_1 error	0.5396	0.3106	0.1348	0.0636	0.0390	0.0080	0.0035



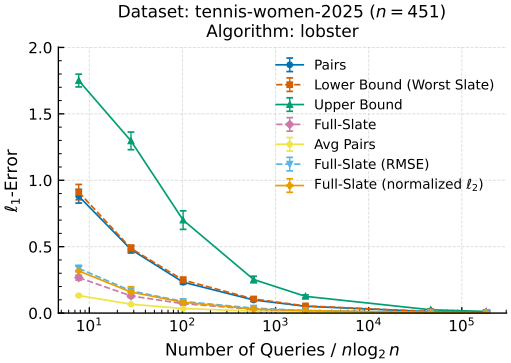
(a) Results for dataset `tennis-women` and algorithm ILSR



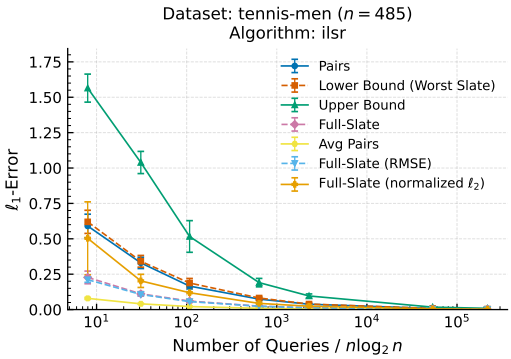
(b) Results for dataset `tennis-women` and algorithm Lobster



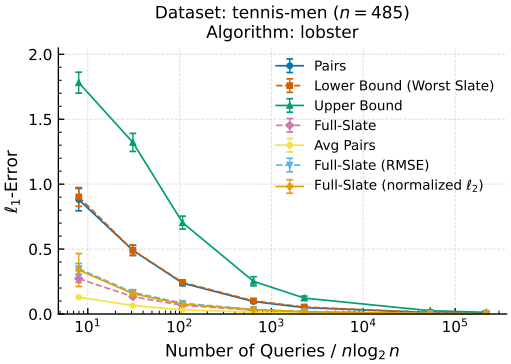
(c) Results for dataset `tennis-women-2025` and algorithm ILSR



(d) Results for dataset `tennis-women-2025` and algorithm Lobster

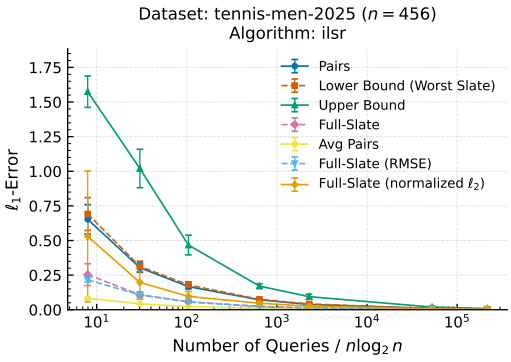


(e) Results for dataset `tennis-men` and algorithm ILSR

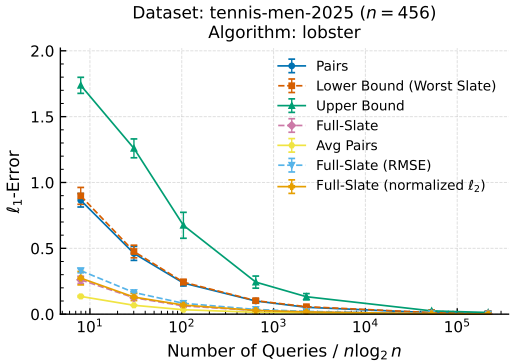


(f) Results for dataset `tennis-men` and algorithm Lobster

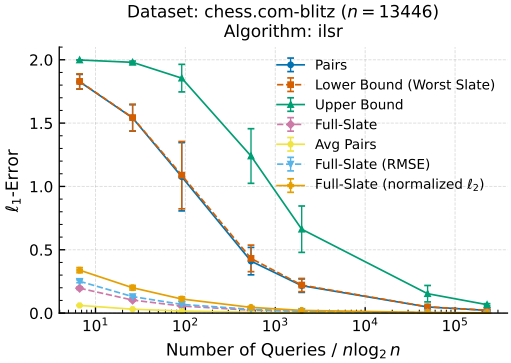
Figure 3: Same setting of Figure 1 but for Tennis datasets and ILSR and Lobster algorithms. In addition to the distances of Figure 1, we also report the RMSE $\frac{1}{\sqrt{n}}\|\theta - \theta'\|_2$, and the normalized ℓ_2 error $\frac{1}{\sqrt{n}} \cdot \frac{\|M_{[n]} - M'_{[n]}\|_2}{\|M_{[n]}\|_2}$.



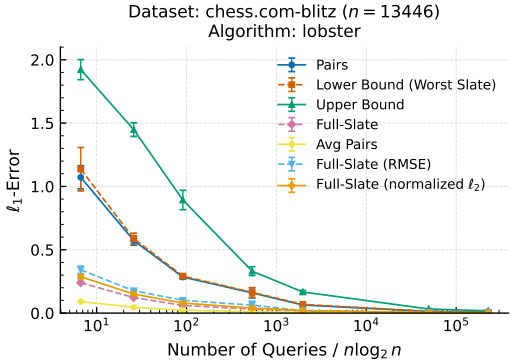
(a) Results for dataset `tennis-men-2025` and algorithm ILSR



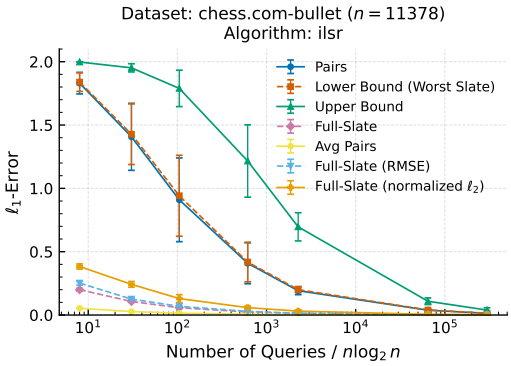
(b) Results for dataset `tennis-men-2025` and algorithm Lobster



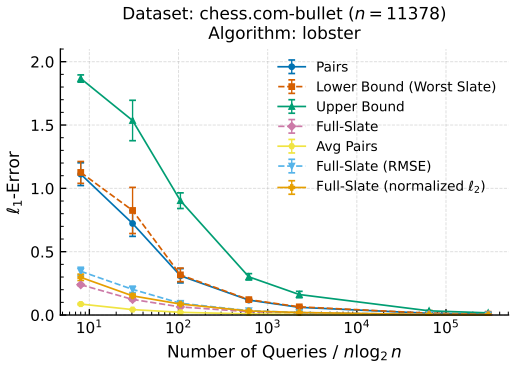
(c) Results for dataset `chess-blitz` and algorithm ILSR



(d) Results for dataset `chess-blitz` and algorithm Lobster

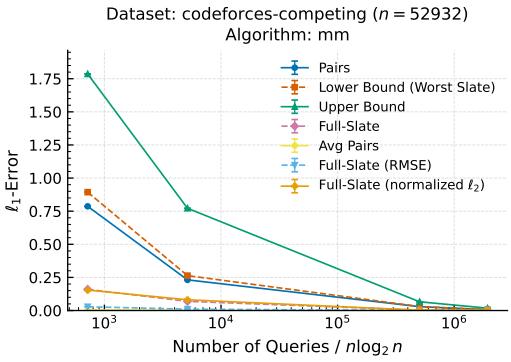


(e) Results for dataset `chess-bullet` and algorithm ILSR

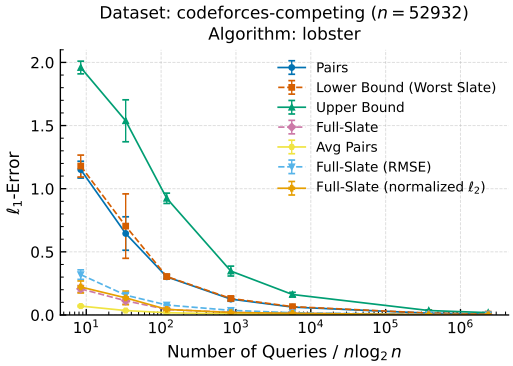


(f) Results for dataset `chess-bullet` and algorithm Lobster

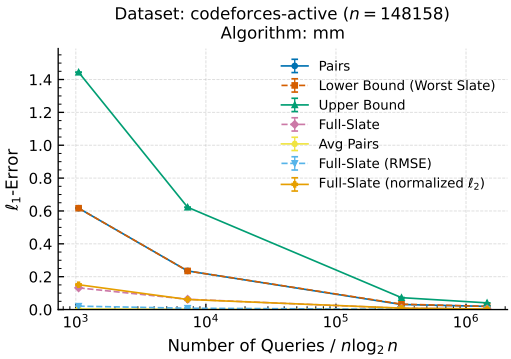
Figure 4: Same setting of Figure 3 but for Tennis and Chess datasets.



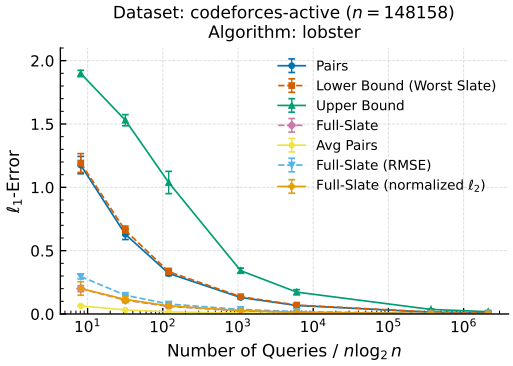
(a) Results for dataset `codeforces-competing` and algorithm MM



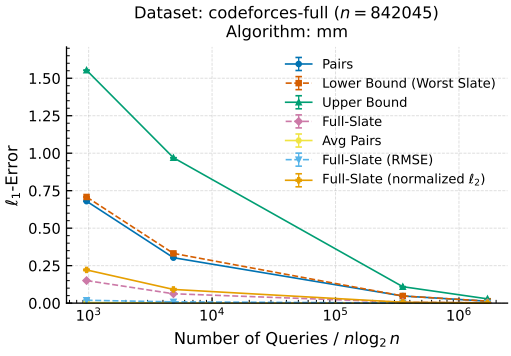
(b) Results for dataset `codeforces-competing` and algorithm Lobster



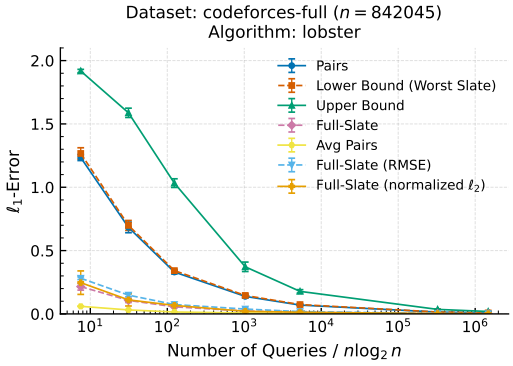
(c) Results for dataset `codeforces-active` and algorithm MM



(d) Results for dataset `codeforces-active` and algorithm Lobster

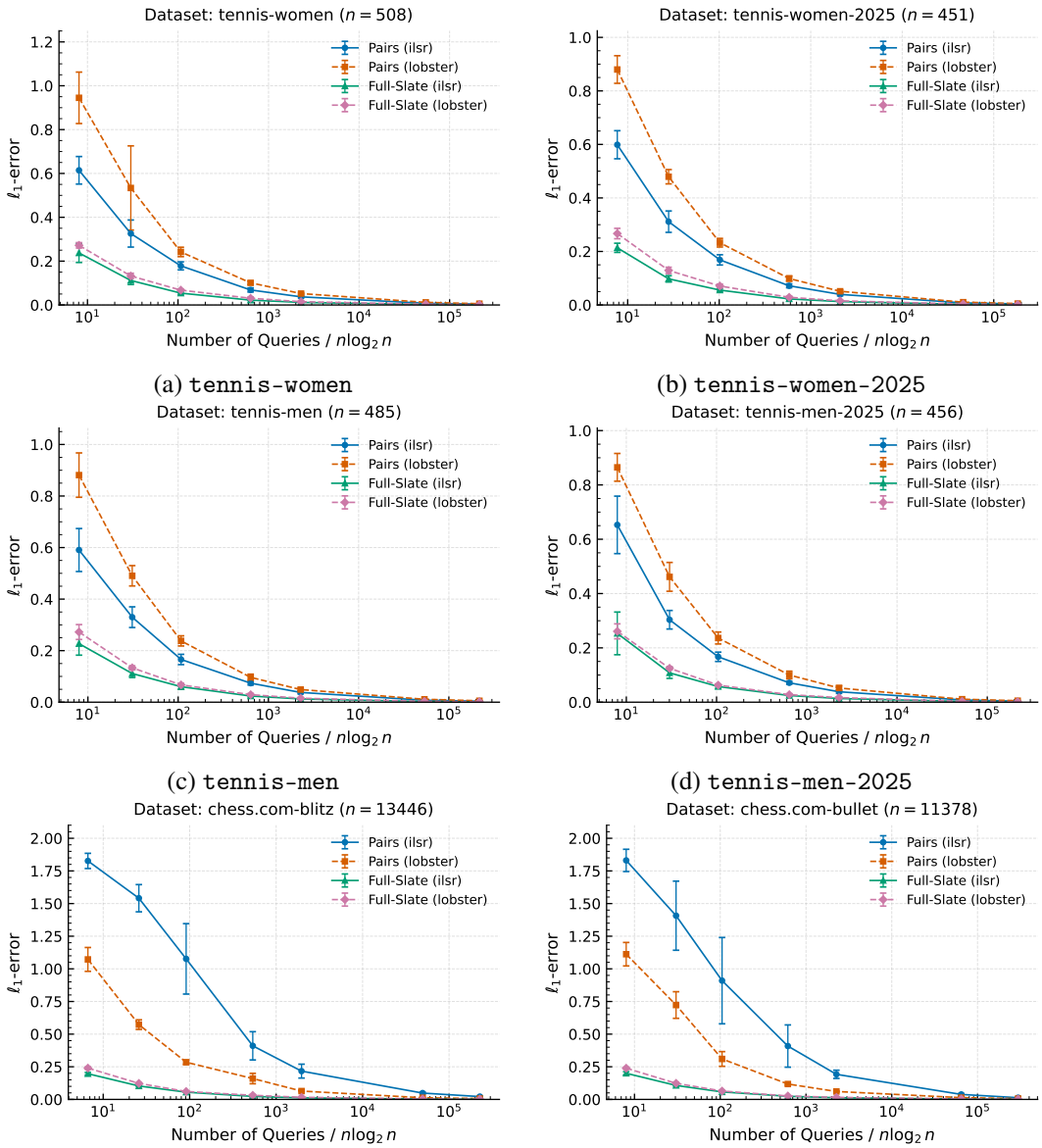


(e) Results for dataset `codeforces-full` and algorithm MM



(f) Results for dataset `codeforces-full` and algorithm Lobster

Figure 5: Same setting of Figure 3 but for Codeforces datasets. Note that we used MM instead of ILSR because ILSR required too much memory. We run MM starting with $10^3 \cdot n \log_2 n$ queries for computational reasons.

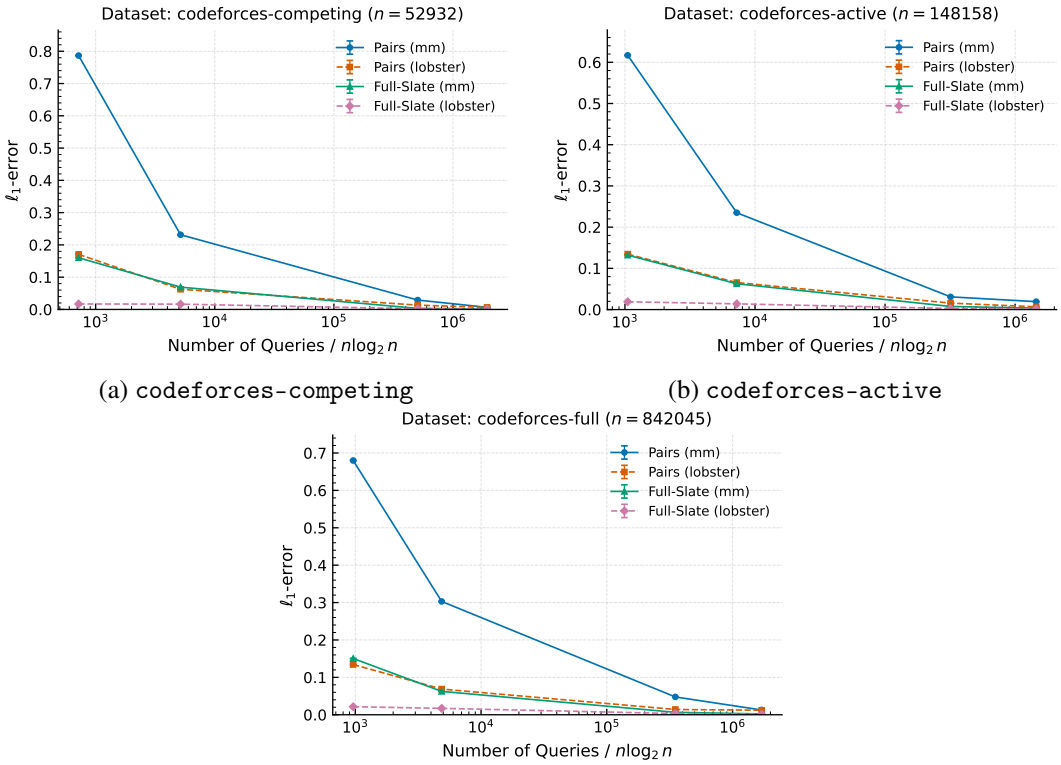


(a) `tennis-women` (b) `tennis-women-2025`

(c) `tennis-men` (d) `tennis-men-2025`

(e) `chess-blitz` (f) `chess-bullet`

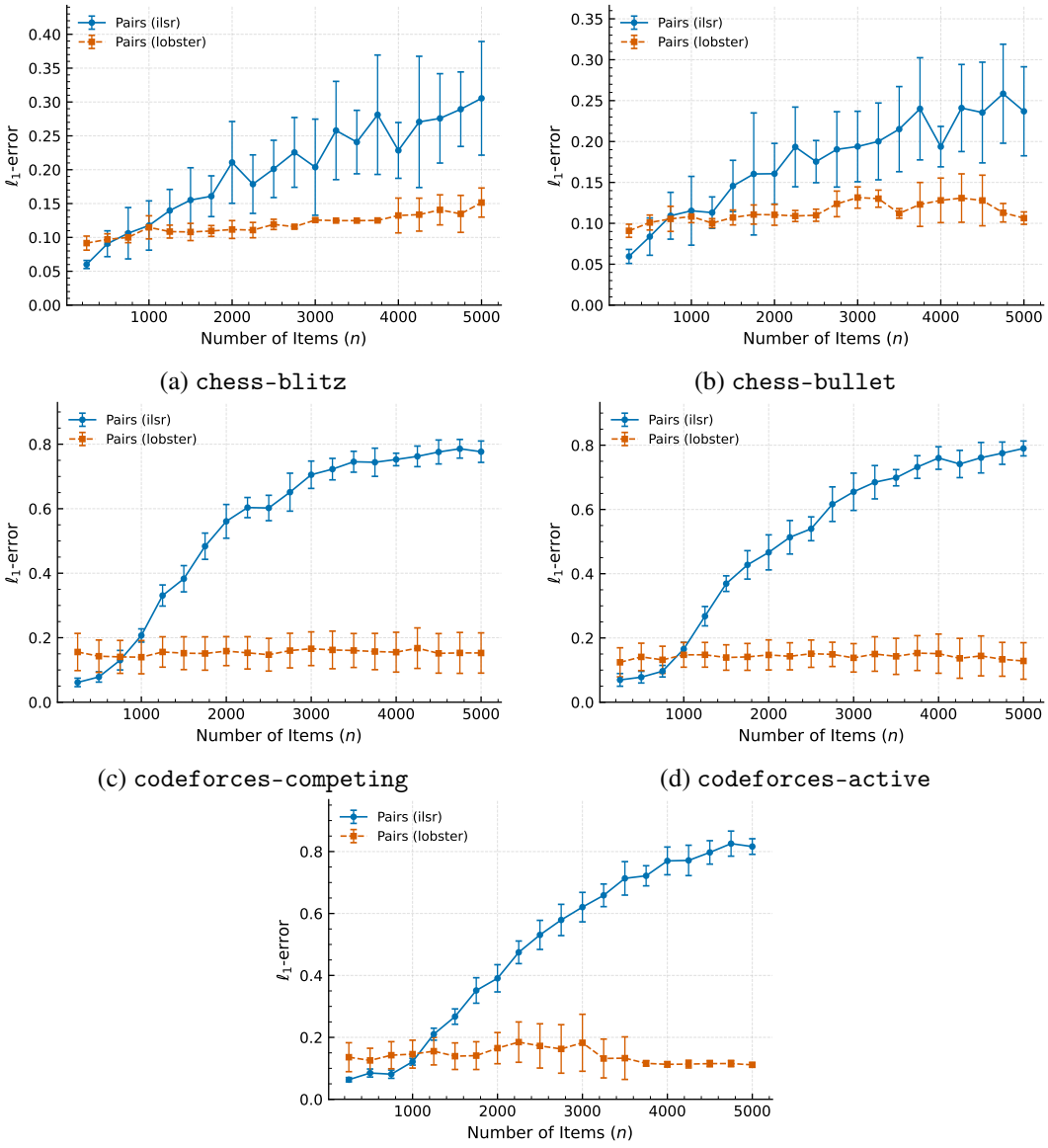
Figure 6: Same setting of Figure 2 but for the Tennis and Chess datasets.



(a) `codeforces-competing` (b) `codeforces-active`

(c) `codeforces-full`

Figure 7: Same setting of Figure 2 but for the Codeforces datasets. Note that we use MM instead of ILSR as the non-adaptive algorithm because ILSR required too much memory. We run MM starting with $10^3 \cdot n \log_2 n$ queries for computational reasons.

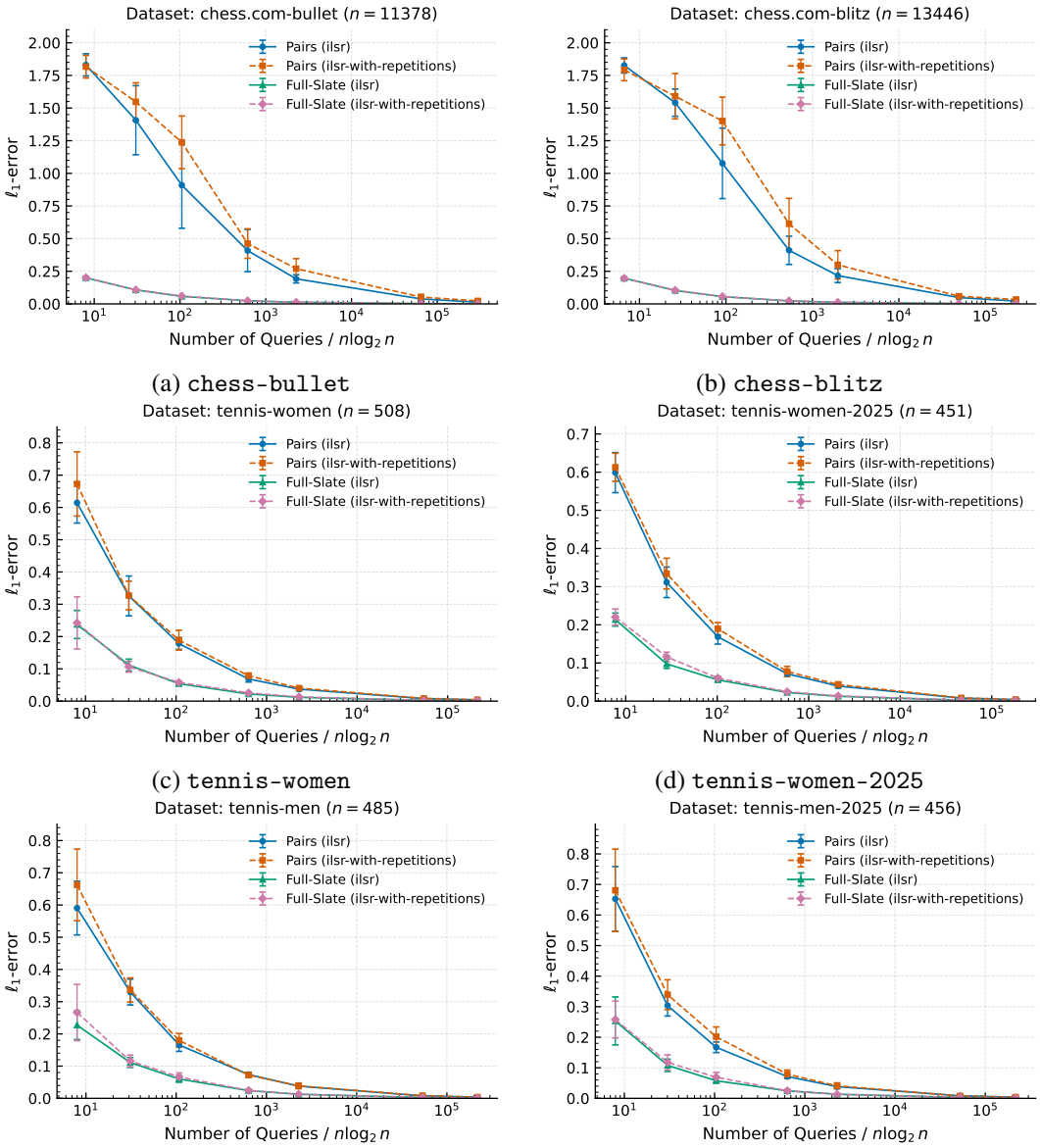


(a) `chess-blitz` (b) `chess-bullet`

(c) `codeforces-competing` (d) `codeforces-active`

(e) `codeforces-full`

Figure 8: Same setting as Figure 2, but we vary the number of items in the dataset by sparsifying them. The x-axis indicates the number of items. The y-axis is the ℓ_1-error on pairs. We use $\varepsilon = 0.1$ to run Lobster, which corresponds roughly to $10^3 \cdot n \log_2 n$ queries; we then run ILSR with the same query budget that we distribute uniformly at random over pairs.

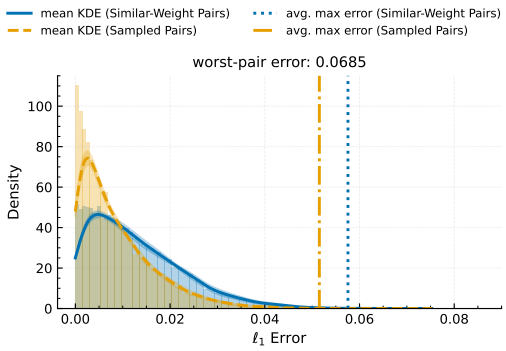


(a) `chess-bullet` (b) `chess-blitz`

(c) `tennis-women` (d) `tennis-women-2025`

(e) `tennis-men` (f) `tennis-men-2025`

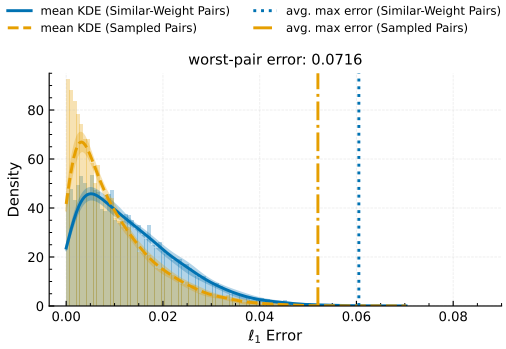
Figure 9: Same setting as Figure 2, but we compare ILSR with uniform at random sampling, and ILSR where we first sample $10 \cdot n \ln n$ pairs and then distribute the remaining budget roughly uniformly among such pairs (ilsr-with-repetitions).



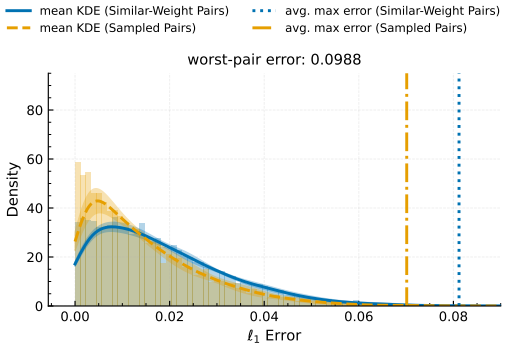
(a) Results for dataset `tennis-women` and algorithm ILSR



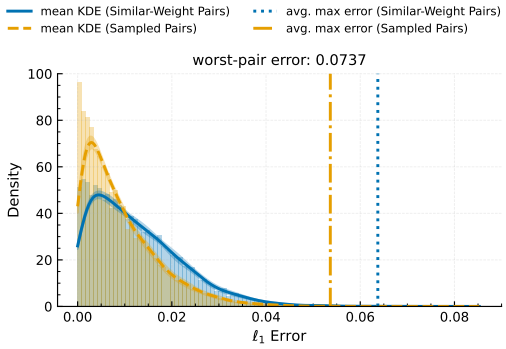
(b) Results for dataset `tennis-women` and algorithm Lobster



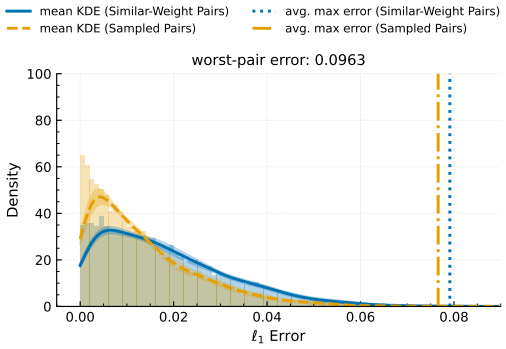
(c) Results for dataset `tennis-women-2025` and algorithm ILSR



(d) Results for dataset `tennis-women-2025` and algorithm Lobster

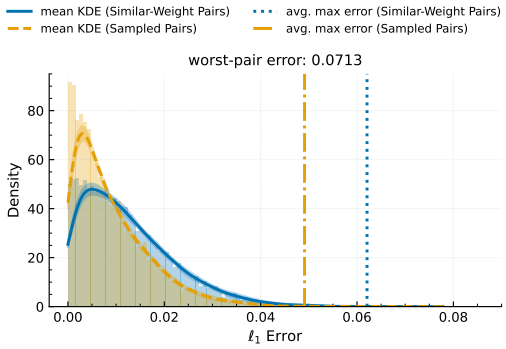


(e) Results for dataset `tennis-men` and algorithm ILSR

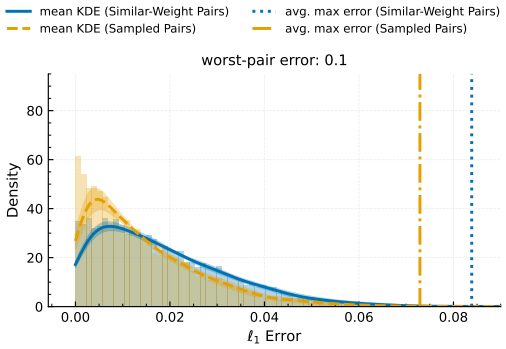


(f) Results for dataset `tennis-men` and algorithm Lobster

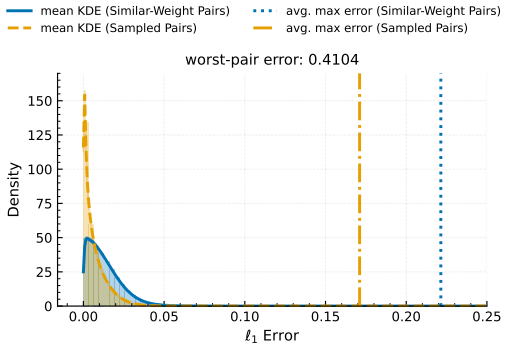
Figure 10: Error distribution on pairs of items of similar weights, compared to pairs sampled uniformly at random. We report the Gaussian Kernel Density Estimate (KDE) of the distribution and also the histogram (shaded area) with 100 buckets. Results are averaged over 10 runs.



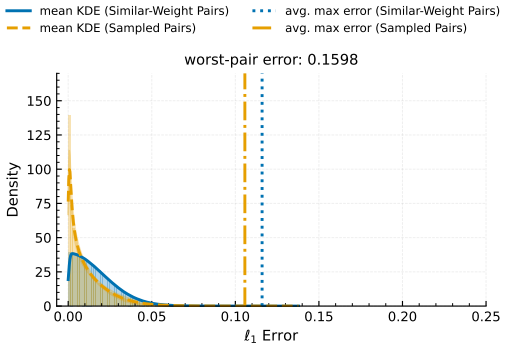
(a) Results for dataset `tennis-men-2025` and algorithm ILSR



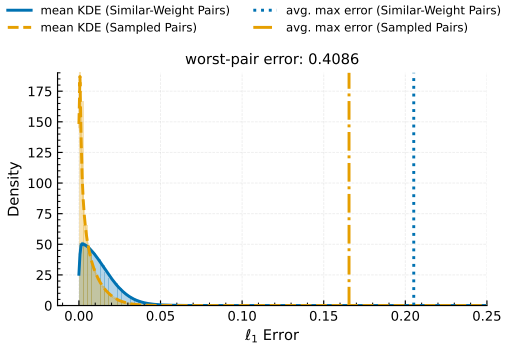
(b) Results for dataset `tennis-men-2025` and algorithm Lobster



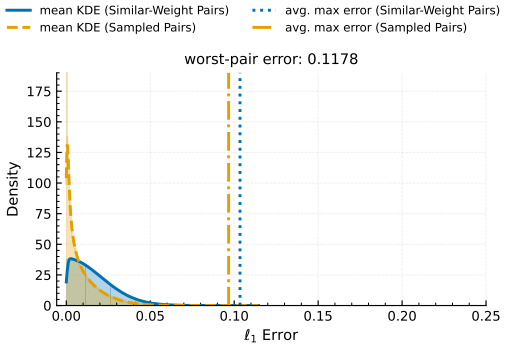
(c) Results for dataset `chess-blitz` and algorithm ILSR



(d) Results for dataset `chess-blitz` and algorithm Lobster

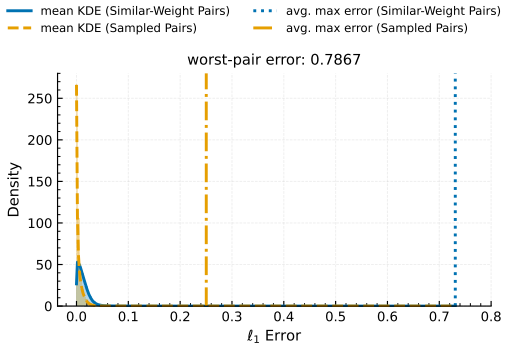


(e) Results for dataset `chess-bullet` and algorithm ILSR

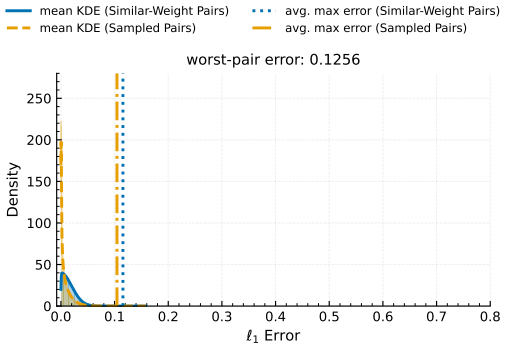


(f) Results for dataset `chess-bullet` and algorithm Lobster

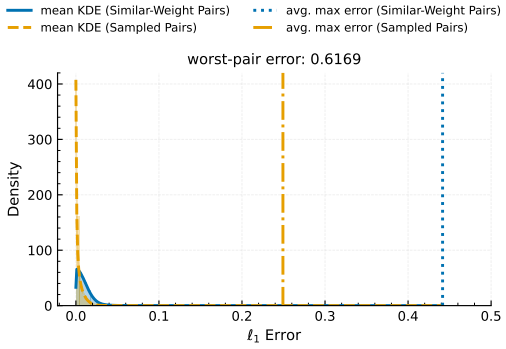
Figure 11: Same setting of Figure 10 for other datasets.



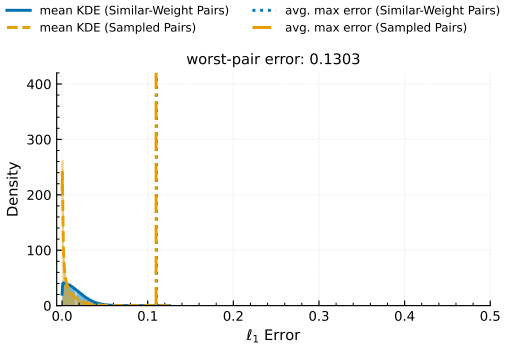
(a) Results for dataset `codeforces-competing` and algorithm MM



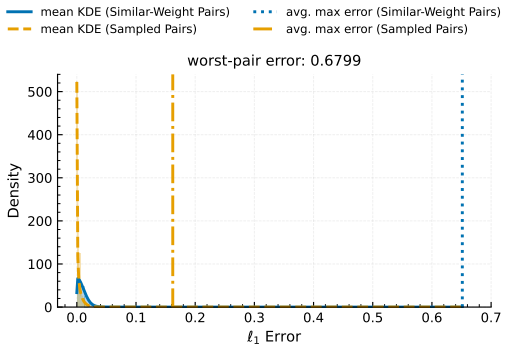
(b) Results for dataset `codeforces-competing` and algorithm Lobster



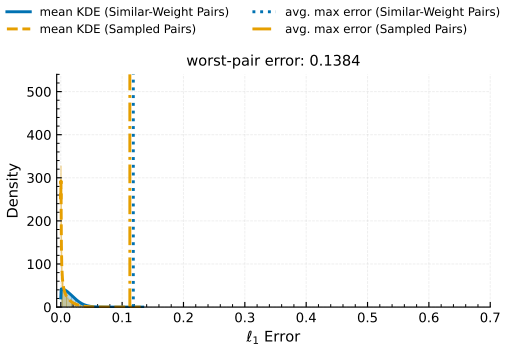
(c) Results for dataset `codeforces-active` and algorithm MM



(d) Results for dataset `codeforces-active` and algorithm Lobster

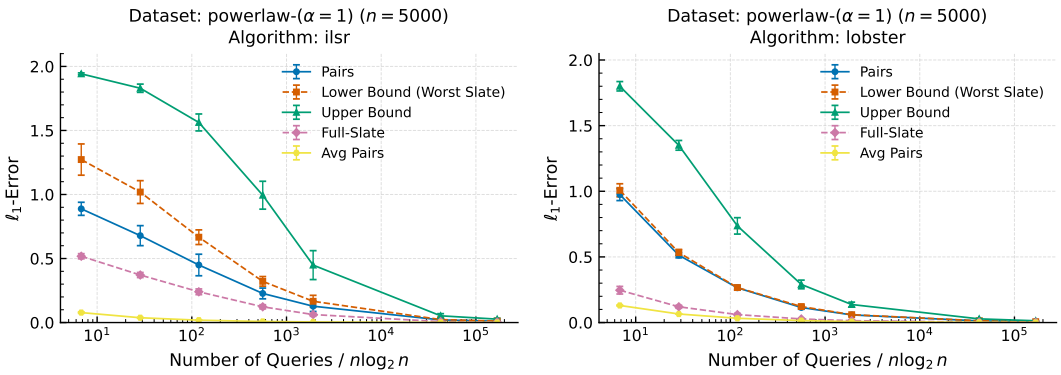


(e) Results for dataset `codeforces-full` and algorithm MM



(f) Results for dataset `codeforces-full` and algorithm Lobster

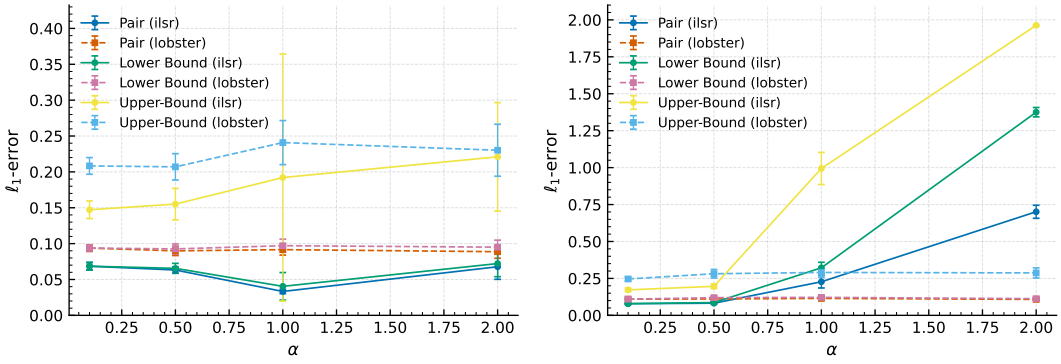
Figure 12: Same setting of Figure 10 for other datasets.



(a) Results for ILSR.

(b) Results for Lobster.

Figure 13: Same setting as Figure 1, but for the synthetic power-law dataset with $w_i \propto i^{-\alpha}$, with $n = 5000$ and $\alpha = 1$.



(a) Results for $n = 250$.

(b) Results for $n = 5000$.

Figure 14: Same setting as Figure 2, but for the synthetic power-law dataset and we vary $\alpha \in \{0.1, 0.5, 1, 2\}$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The author(s) have ensured that the contents of the abstract and introduction are a faithful representation of the contributions provided in the papers. The theoretical claims made at the beginning of the paper are substantiated by formal proofs in the body and appendices; the empirical claims are justified by experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a discussion of limitations of our work in a dedicated paragraph in Section 5. Due to space constraints, we did not make this a fully self-standing section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All of the theoretical results presented come with clearly stated assumptions, as well as thorough theoretical proofs. The proofs that did not fit in the main body are included in the appendices.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In the sections and appendices describing the experiments carried out (Section 3 and Appendix J.1) the reader will find a detailed explanation of the experimental setup and the choice of (hyper)parameters made. The code used for the experiments is also included in the submission.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We include the code as well as the dataset used to run all the experiments. We will make the code public should the paper be accepted.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: These are discussed in the experimental sections.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The plots include error bars, representing the standard deviations of the quantities plotted.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This information can be found in Appendix J.1. It is worth noting that the experiments focus on error per number of queries (rather than, say, wall-clock time) and hence the results plotted should not depend on the specific machine used to run them.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Ethics Guidelines reviewed and followed.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: The work presented in this paper is largely foundational. We did not identify any direct path to negative applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: No new datasets/models released by this work. Datasets used carry no safety risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The libraries we used are properly credited. The data is sourced from appropriately mentioned APIs.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets were introduced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human study subjects are involved in the experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human study subjects are involved in the experiments.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: In Appendix B and Appendix I we made use of LLM to aid with finding proofs and algorithms; this is discussed in those sections. All LLM contributions were verified by the authors for correctness and in some cases, were simplified and/or rewritten.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.