Approximating a RUM from Distributions on k-Slates

Flavio Chierichetti
Sapienza University
Dip. di Informatica

Mirko Giacchini
Sapienza University
Dip. di Informatica

Ravi Kumar
Google
Mountain View, CA

Alessandro Panconesi
Sapienza University
Dip. di Informatica

Andrew Tomkins
Google
Mountain View, CA

Abstract

In this work we consider the problem of fitting Random Utility Models (RUMs) to user choices. Given the winner distributions of the subsets of size k of a universe, we obtain a polynomial-time algorithm that finds the RUM that best approximates the given distribution on average. Our algorithm is based on a linear program that we solve using the ellipsoid method. Given that its corresponding separation oracle problem is NP-hard, we devise an approximate separation oracle that can be viewed as a generalization of the weighted feedback arc set problem to hypergraphs. Our theoretical result can also be made practical: we obtain a heuristic that is effective and scales to real-world datasets.

1 INTRODUCTION

In this paper we consider the setting of discrete choice, in which users must choose an item from a set of alternatives. The gold standard for highly representative models of discrete choice is the family of *Random Utility Models*, or *RUMs*, which are capable of encoding a range of complex outcomes under a broad notion of rational user behavior. RUMs are quite powerful but unfortunately there are no known algorithms guaranteed to learn or approximate a RUM from samples of user choices.

It is therefore of great interest to find special cases for which tractable algorithms are possible. Recent work of Almanza et al. (2022) showed that efficient algorithms for approximate learning of general RUMs are possible in the *pairwise* setting—if the user choices being modeled contain only two options.

There are some settings in which pairwise choices arise naturally, such as head-to-head testing of two items from a universe, or two-player games. But most practical settings involve choices among larger sets of options. Many online platforms offer interfaces with multiple options to choose from: ten blue links, or a fixed set of movie options shown in a carousel. In this paper, we extend earlier results to give a polynomial-time algorithm to approximately learn RUMs on slates of size at most k, for any constant k. Formal definitions for approximate learning of RUMs on slates of size at most k are given in Section 2.

To describe the motivation and the idea behind the algorithm, we must first introduce the idea of RUMs as distributions over permutations over a universe of items $[n]$.When a user is presented with a slate of objects, a random permutation is drawn, and the object selected is the one of highest rank in the permutation. Note that a RUM, for each slate S, induces a probability distribution over S, specifying the probability that $i \in S$ is selected by the user when S is given. In this context, there are two natural learning tasks to consider. The input to both tasks consists of a set of pairs (i, S) where $i \in S$ is the object selected by the user when S is presented. With this information one can compute the empirical probability that an object is selected from a given slate. The first learning task is to compute a RUM whose induced probability distributions best approximate the empirical distributions. The second learning task is, given a set of past such interactions, to predict the empirical probability in the future.

Our Contributions. In this paper we give, for both learning tasks, *(i)* a polynomial-time algorithm for slates of size $k \geq 2$, thereby significantly extending the results in Almanza et al. (2022), and *(ii)* fast heuristics that are effective in practice.

As in Almanza et al. (2022), we use the celebrated ellipsoid method for linear programming (LP), together with a separation oracle. Unfortunately, their approach only works only for the case of slates of size 2, i.e., for $k = 2$. The ellipsoid method, as it is well-known, allows to cope with LPs with exponentially many constraints, provided that a so-called separation oracle is available. The technical difficulty here is to exhibit such a separation oracle for the case of slates of size $k > 2$. In particular, the separation oracle

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

can no longer be formed by solving the minimum *feedback arc set* problem (FAS) as in Almanza et al. (2022). Instead, we require a solution to a new problem that represents a hypergraph extension of FAS. The main contribution of this work is defining the new extended problem, giving a polynomial-time algorithm that produces an approximate solution, and then showing that the resulting approximate separation oracle provides a sufficient approximation to the best possible RUM.

To conclude, we would like to mention a couple of positive features of our approach. Our polynomial-time solution requires information about all slates of dimension k in order to work, but such complete datasets might not be available in practice. We thus provide a heuristic that is able to cope effectively with such contingencies. We also provide heuristic but practically more efficient separation oracles, which are described in the experimental section.

Previous Work. Discrete choice theory is a well-established research topic in machine learning and economics; see (Train, 2003) for an excellent introduction and (Chierichetti et al., 2018a, 2021; Rosenfeld et al., 2020; Seshadri et al., 2020, 2019) for some recent work. RUMs are a general class that contains, e.g., Multinomial Logits (or MNLs) models and Mixed MNLs (McFadden and Train, 2000; Chierichetti et al., 2021), as special cases. RUMs are equivalent (Chierichetti et al., 2018b; Farias et al., 2009) to models in which a user samples an ordering of the objects in the universe, and given a slate, selects the object in the slate ranked the highest in the ordering.

RUMs, and subclasses of RUMs, have also been extensively studied from both active learning and passive learning perspectives (Soufiani et al., 2012; Oh and Shah, 2014; Chierichetti et al., 2018b,a; Negahban et al., 2018; Tang, 2020). The problems of efficiently representing and sketching RUMs have been studied (Farias et al., 2009; Chierichetti et al., 2021; Almanza et al., 2022).

The work that is closest to ours is (Almanza et al., 2022), which also aims to learn RUMs from choices on slates by solving an LP via the ellipsoid method with an approximate separation oracle. The differences between our work and theirs can be summarized as follows: (i) their algorithm works *only* for slates of size $k = 2$, while ours works for any constant k, (ii) their LP also works *only* for $k = 2$ and, most importantly, (iii) our separation oracles are *different* and *new*: we (approximately) solve a novel generalization of the FAS problem; this might be of independent interest.

Organization. Section 2 establishes the notation. Section 3 provides formulations of the primal and dual LPs for RUM fitting, Section 4 introduces the pivotal element of the approximate separation oracle for the LP, Section 5 shows that this oracle can be used with the ellipsoid method to find an approximately optimal RUM. Section 6 discusses lower bounds for the size of the input for our LP method. Section 7 presents our experimental results.

2 PRELIMINARIES

For a set S, let 2^S denote the set of all subsets of S and let $\binom{S}{k}$ denote the set of all k-sized subsets of S. For a distribution D, let $x \sim D$ denote that the random variable x is drawn according to D and let $D(i)$ denote $\Pr_{x\sim D}[x = i]$, where i is in the support of D.

Let $[n]$ denote $\{1, \ldots, n\}$ and let $\mathbf{S}_n$ denote the set of permutations of $[n]$. A *slate* is a non-empty subset of $[n]$. For a slate $\varnothing \neq S \in 2^{[n]}$ and a permutation $\pi \in \mathbf{S}_n$, let $\pi(S)$ be the element of S that ranks the highest in π.

Definition 1 (Random utility model (RUM))**.** *A* random utility model (RUM) R *on* $[n]$ *is a distribution* D *on* $\mathbf{S}_n$*. For a slate* S*, let* R_S *denote the distribution of the random variable* $\pi(S)$*, where* $\pi \sim D$*. We say that* R_S *is the* winner distribution *on* S *induced by RUM* R*.*

Our goal is to fit RUMs to datasets in order to minimize the average ℓ_1-error over the winner distributions.

Definition 2 (Average RUM approximation)**.** *Let* $\mathcal{S}$ *be a set of slates of* $[n]$ *and for each* $S \in \mathcal{S}$*, let* $\mathcal{P}(S)$ *be a probability distribution over* S*. We say that* $\mathcal{P}$ *can be* approximated on average *to within* ϵ *by a RUM* R *if* $\operatorname{avg}_{S\in\mathcal{S}} |R_S - \mathcal{P}(S)|_1 \leq \epsilon$*. Given* $\mathcal{P}$*, let* $\epsilon_1(\mathcal{P})$ *be the smallest*[1] $x \geq 0$ *such that there is a RUM that approximates* $\mathcal{P}$ *on average to within* x*.*

Recall that the ℓ_1-distance between two distributions over S is (exactly) twice as large as the total variation distance between them, i.e., it is twice as large as the maximum gap in the probabilities of an event in the two distributions.

Problem 3 (Average RUM additive approximation)**.** *Given* $\mathcal{S}$ *and a corresponding* $\mathcal{P}$*, find a* δ-additive approximation *to* $\epsilon_1(\mathcal{P})$*, i.e., obtain a RUM whose average distance from* $\mathcal{P}$ *is not larger than* $\epsilon_1(\mathcal{P}) + \delta$*.*

We also need the following generalization of the weighted feedback edge set problem to hypergraphs. For a hyperedge $e = (x_1, \ldots, x_k) \in \binom{[n]}{k}$ and a permutation $\pi \in \mathbf{S}_n$, let $\pi(e) = (\pi(x_1), \ldots, \pi(x_k))$.

Problem 4 (Weighted feedback hyperedge set (WFHS))**.** *An instance of the* (τ, k)-bounded weighted feedback hyperedge set (WFHS) *problem is composed of a set* $V = [n]$ *of vertices, a set* $E \subseteq \binom{V}{k}$ *of hyperedges and, for each* $e \in E$*, a non-negative weight function* $w_e : e \to [0, \tau]$*. The cost of a permutation* π *of the vertices of* V *is equal to* $C(\pi) = \sum_{e\in E} w_e(\pi(e))$*. The WFHS problem is to find a permutation* $\pi \in \mathbf{S}_n$ *that minimizes* $C(\pi)$*.*

[1] This minimum exists since it is the optimal value of a finite-sized, feasible, LP.

3 FITTING RUMS WITH LINEAR PROGRAMS

Our goal in this section is to write down a linear program (LP) whose solution gives the desired RUM. This LP will have exponentially many constraints but polynomially many variables. Thanks to the general theory of the ellipsoid method, it is possible to solve the LP to within a very small error, provided that a so-called *separation oracle* exists. Such an oracle is provided in the next sections.

Let us begin by writing down an LP whose solution gives a RUM attaining minimum average ℓ_1-error for the slates in $\mathcal{S}$, and which generalizes the LP of Almanza et al. (2022) to $k \geq 2$.[2] In order to write it down, we need to have access to $\mathcal{P}$. For simplicity, let D_S denote $\mathcal{P}(S)$. Then $D_S(i)$ is the (empirical) probability that i wins in S, for each $i \in S \in \mathcal{S}$. (Observe that, for each $S \in \mathcal{S}$, $\sum_{i \in S} D_S(i) = 1$.) The LP assigns probability $p_\pi \geq 0$ to each permutation $\pi \in \mathbf{S}_n$, and requires that $\sum_{\pi \in \mathbf{S}_n} p_\pi = 1$.

$$\begin{cases} \min \frac{1}{|\mathcal{S}|} \cdot \sum_{S \in \mathcal{S}} \sum_{i \in S} \epsilon_{S,i} & & \\ \epsilon_{S,i} + \sum_{\substack{\pi \in \mathbf{S}_n \\ \pi(S)=i}} p_\pi \geq D_S(i) & (L_{S,i}) & \forall i \in S \in \mathcal{S} \\ \epsilon_{S,i} - \sum_{\substack{\pi \in \mathbf{S}_n \\ \pi(S)=i}} p_\pi \geq -D_S(i) & (U_{S,i}) & \forall i \in S \in \mathcal{S} \\ \sum_{\pi \in \mathbf{S}_n} p_\pi = 1 & (D) & \\ p_\pi \geq 0 & & \forall \pi \in \mathbf{S}_n \\ \epsilon_{S,i} \geq 0 & & \forall i \in S \in \mathcal{S} \end{cases} \tag{1}$$

Then, $\{p_\pi\}_{\pi \in \mathbf{S}_n}$ defines a RUM. Given any $i \in S \in \mathcal{S}$, it also requires that the approximation error made by every RUM that is a feasible solution for the pair (i, S) is no greater than $\epsilon_{S,i}$. The constraints of type $L_{S,i}$ guarantee that the probability that i wins in S is at least $D_S(i) - \epsilon_{S,i}$; those of type $U_{S,i}$ guarantee that the same probability is at most $D_S(i) + \epsilon_{S,i}$. Therefore, the ℓ_1-error made by the optimal RUM on slate S is not more than $\sum_{i \in S} \epsilon_{S,i}$. The LP minimizes the average (i.e., the scaled sum) of the ℓ_1-errors over all slates $\mathcal{S}$ (i.e., $|\mathcal{S}|^{-1} \sum_{S \in \mathcal{S}} \sum_{i \in S} \epsilon_{S,i}$). It follows that the optimal solution ensures that each $\epsilon_{S,i}$ *equals* the ℓ_1-error on S and therefore that its average ℓ_1-error is the minimum achievable by any RUM.

LP (1) has exponentially many variables and polynomially many constraints. If we take its dual, we obtain an LP with polynomially many variables and exponentially many constraints. Hence, it can be optimized efficiently, given a separation oracle. Here is the dual of (1):

$$\begin{cases} \max D + \sum_{i \in S \in \mathcal{S}} (D_S(i) \cdot (L_{S,i} - U_{S,i})) & & \\ L_{S,i} + U_{S,i} \leq |\mathcal{S}|^{-1} & (\epsilon_{S,i}) & \forall i \in S \in \mathcal{S} \\ D + \sum_{S \in \mathcal{S}} (L_{S,\pi(S)} - U_{S,\pi(S)}) \leq 0 & (p_\pi) & \forall \pi \in \mathbf{S}_n \\ L_{S,i}, U_{S,i} \geq 0 & & \\ D \text{ unrestricted} & & \end{cases}$$

Observe that every feasible dual solution can be transformed into a feasible solution with the same value and with the additional property that, for all $i \in S \in \mathcal{S}$, at least one of $L_{S,i}$ and $U_{S,i}$ is equal to zero. To see this, observe that if the two variables are positive we can subtract $\min(L_{S,i}, U_{S,i})$ from both without affecting feasibility and without changing the objective function's value. Given a feasible solution to the dual, define $\Delta_{S,i} = U_{S,i} - L_{S,i}$. With this transformation, we have $U_{S,i} = \max(\Delta_{S,i}, 0)$, $L_{S,i} = \max(-\Delta_{S,i}, 0)$, and $|\Delta_{S,i}| = L_{S,i} + U_{S,i}$. The dual LP is then equivalent to the following LP:

$$\begin{cases} \max D - \sum_{i \in S \in \mathcal{S}} (D_S(i) \cdot \Delta_{S,i}) & \\ \sum_{S \in \mathcal{S}} \Delta_{S,\pi(S)} \geq D & \forall \pi \in \mathbf{S}_n \\ -|\mathcal{S}|^{-1} \leq \Delta_{S,i} \leq |\mathcal{S}|^{-1} & \forall i \in S \in \mathcal{S} \\ D \text{ unrestricted} & \end{cases} \tag{2}$$

We now transform LP (2) from a maximization problem into a feasibility problem, to pave the way for the ellipsoid algorithm:

$$F_\rho := \begin{cases} D - \sum_{i \in S \in \mathcal{S}} (D_S(i) \cdot \Delta_{S,i}) \geq \rho & \\ \sum_{S \in \mathcal{S}} \Delta_{S,\pi(S)} \geq D & \forall \pi \in \mathbf{S}_n \\ -|\mathcal{S}|^{-1} \leq \Delta_{S,i} \leq |\mathcal{S}|^{-1} & \forall i \in S \in \mathcal{S} \end{cases}$$

Observe that LP (2) has value at least ρ iff F_ρ is feasible. In order to have the polytope lie in the non-negative orthant, we set $\overline{\Delta}_{S,i} = \Delta_{S,i} + |\mathcal{S}|^{-1}$. We then rewrite the expressions in terms of $\overline{\Delta}_{S,i}$:

$$\begin{aligned} & D - \sum_{i \in S \in \mathcal{S}} (D_S(i) \cdot \Delta_{S,i}) \\ &= D - \sum_{i \in S \in \mathcal{S}} \left(D_S(i) \cdot \left(\overline{\Delta}_{S,i} - |\mathcal{S}|^{-1}\right)\right) \\ &= D - \sum_{i \in S \in \mathcal{S}} \left(D_S(i) \cdot \overline{\Delta}_{S,i}\right) + \sum_{S \in \mathcal{S}} \sum_{i \in S} \left(D_S(i) \cdot |\mathcal{S}|^{-1}\right) \\ &= D - \sum_{i \in S \in \mathcal{S}} \left(D_S(i) \cdot \overline{\Delta}_{S,i}\right) + \sum_{S \in \mathcal{S}} |\mathcal{S}|^{-1} \\ &= D - \sum_{i \in S \in \mathcal{S}} \left(D_S(i) \cdot \overline{\Delta}_{S,i}\right) + 1. \end{aligned}$$

Thus, $D - \sum_{i \in S \in \mathcal{S}} (D_S(i) \cdot \Delta_{S,i}) \geq \rho$ is equivalent to $D - \sum_{i \in S \in \mathcal{S}} \left(D_S(i) \cdot \overline{\Delta}_{S,i}\right) \geq \rho - 1$. Moreover, for

[2] For the interested reader, the LP of Almanza et al. (2022) can be obtained from ours by setting $k = 2$. The main technical difference between the LPs stems from the fact that, when $k = 2$, enforcing a bound on the error of the probability that a particular element wins in a slate is *equivalent* to enforcing a bound on the total variation error of the slate distributions (for distributions $P = (x, 1-x)$ and $Q = (y, 1-y)$ on two elements, it is easy to see that $|P - Q|_1 = 2|P - Q|_\infty$); unfortunately, this property fails to hold if $k > 2$. We thus have to provide an argument that makes use of a novel polyhedron ($\overline{F}_\rho$).

$\pi \in \mathbf{S}_n$, $\sum_{S\in\mathcal{S}} \Delta_{S,\pi(S)} = \sum_{S\in\mathcal{S}} \left(\overline{\Delta}_{S,\pi(S)} - |\mathcal{S}|^{-1}\right) = \sum_{S\in\mathcal{S}} \overline{\Delta}_{S,\pi(S)} - 1$. Thus, $\sum_{S\in\mathcal{S}} \Delta_{S,\pi(S)} \geq D$ is equivalent to $\sum_{S\in\mathcal{S}} \overline{\Delta}_{S,\pi(S)} \geq D+1$. Finally, $-|\mathcal{S}|^{-1} \leq \Delta_{S,i} \leq |\mathcal{S}|^{-1}$ is equivalent to $0 \leq \overline{\Delta}_{S,i} \leq 2|\mathcal{S}|^{-1}$. Hence, the system F_ρ is equivalent to the following system $\overline{F}_\rho$.

$$\overline{F}_\rho := \begin{cases} c_\rho : D - \sum\limits_{i\in S\in\mathcal{S}} \left(D_S(i)\cdot\overline{\Delta}_{S,i}\right) \geq \rho - 1 \\ c_\pi : \sum\limits_{S\in\mathcal{S}} \overline{\Delta}_{S,\pi(S)} \geq D+1 \quad \forall \pi \in \mathbf{S}_n \\ c_{S,i} : 0 \leq \overline{\Delta}_{S,i} \leq 2|\mathcal{S}|^{-1} \quad \forall i \in S \in \mathcal{S} \end{cases} \tag{3}$$

LP (2) has value at least ρ if and only if $\overline{F}_\rho$ is feasible.

To summarize, we have reduced Problem 3 to the feasibility problem $\overline{F}_\rho$. Since $\overline{F}_\rho$ has exponentially many constraints, we cannot check the validity of its c_π constraints one by one[3]. On the other hand, the feasibility of the full set of the c_π constraints of $\overline{F}_\rho$ can be checked by solving an instance of Problem 4, which is an instance of the WFHS problem with weights given by the $\overline{\Delta}_{S,i}$'s. In the next section, we provide an approximation algorithm for WFHS. And in the subsequent section, we use this approximation algorithm to provide an approximate separation oracle for $\overline{F}_\rho$, which we will then use to solve Problem 3, our original RUM fitting problem, in polynomial time.

4 AN ALGORITHM FOR THE WFHS PROBLEM

We introduce notation and recall some results before discussing our $O(\epsilon\cdot\tau\cdot n^k)$-additive approximation algorithm for the WFHS problem; this algorithm, like the algorithms of Kenyon-Mathieu and Schudy (2007); Schudy (2012); Frieze and Kannan (1999) for the feedback arc set (FAS) problem, makes use of tensor-maximization algorithms for k-CSP (constraint satisfaction problem) as a building block.

Our k-CSP will have a constant-sized alphabet $[t]$, for $t = O(\epsilon^{-1})$. The k-CSP has one type of predicate P of arity k: $P(x_1,\ldots,x_k) = [x_1 > \max(x_2,\ldots,x_k)]$.

An instance $\mathcal{I}$ of this *k-CSP* is composed of a set $X = \{x_1,\ldots,x_n\}$ of variables, of a set M of k-tuples of variables of X (the constraints induced by P), and of a weighting $w : M \to [0,\tau]$. The variables take values over $[t]$.

The goal of the problem is to assign values to the variables (where the value of $x_i \in X$ is an element of $[t]$) in order to maximize the total weight of k-tuples $(x_{i_1},\ldots,x_{i_k}) \in M$ that make the predicate $P(x_{i_1},\ldots,x_{i_k})$ true (i.e., maximize $\sum_{(x_{i_1},\ldots,x_{i_k})\in M} \left(P(x_{i_1},\ldots,x_{i_k})\cdot w(x_{i_1},\ldots,x_{i_k})\right)$). This maximum value is called the *optimal value* of the k-CSP instance and is denoted $\mathrm{OPT}(\mathcal{I})$.

[3]The number of $C_{S,i}$'s constraints and c_ρ is only $O(|\mathcal{S}|)$, i.e., it is (sub)linear in the input size, and hence can be easily checked.

We will make use of the following result, that has been proved by various authors including, e.g., (Schudy, 2012; Yaroslavtsev, 2014).

Theorem 5. *For each constant $\epsilon > 0$, for each non-negative integers $t, k \geq 1$, and for each k-CSP over the alphabet $[t]$, there exists an algorithm that, in time $O(n^k)$, returns an assignment to the n variables of a generic instance $\mathcal{I}$ of the k-CSP such that the expected weight of the k-tuples satisfied by the assignment is at least $\mathrm{OPT}(\mathcal{I}) - \epsilon\cdot\tau\cdot n^k$.*

We also mention that a slower algorithm, with a runtime $n^{2^{O(k)}}$, for approximately solving generic k-CSPs was given in Yoshida and Zhou (2014).

We now show that the WFHS problem can be additively approximated. Our approach, like that of Kenyon-Mathieu and Schudy (2007); Schudy (2012); Frieze and Kannan (1999) for the feedback arc set (FAS) problem, solves the WFHS problem by first (i) casting it as a tensor-maximization/k-CSP problem and then (ii) transforming the solution to this problem into an ordering of the vertices.

Theorem 6. *For each constants $0 < \epsilon < \frac{1}{2}, \alpha > 0, k \geq 2$, there exists an algorithm for the (τ,k)-bounded WFHS problem that can return an $(\epsilon\cdot\tau\cdot n^k)$-additive approximation, with probability at least $1 - n^{-\alpha}$, in time $O(\alpha n^k \log n)$.*

Proof. Given an instance $V, E, \{w_e\}_{e\in E}$ of the WFHS problem, we define a k-CSP with variables $x_1,\ldots,x_n$ over the alphabet $[t]$, for $t = \lceil\epsilon^{-1}\rceil$. For each hyperedge $e = \{x_{i_1},\ldots,x_{i_k}\} \in E$, we create k constraints: $C_{i_1,e} := P(x_{i_1},x_{i_2},\ldots,x_{i_{|e|-1}},x_{i_k})$, $C_{i_2,e} := P(x_{i_2},x_{i_3},\ldots,x_{i_{k-1}},x_{i_k},x_{i_1})$, …, $C_{i_k,e} := P(x_{i_k},x_{i_1},\ldots,x_{i_{k-2}},x_{i_{k-1}})$. The weight of the constraint $C_{i_j,e}$ will be $w\left(C_{i_j,e}\right) = \tau - w_e(i_j)$ and let $\Gamma_e = \{C_{i_j,e} \mid i_j \in e\}$. Let $\mathcal{I}$ be the resulting k-CSP instance.

Now, let $\pi \in \mathbf{S}_n$. For each $i \in [t]$, and for each item j having rank in π in the set $R_i = \left[\left\lceil i\cdot\frac{n}{t}\right\rceil\right] \setminus \left[\left\lceil (i-1)\cdot\frac{n}{t}\right\rceil\right]$ (then, $|R_i| \leq \frac{n}{t}+1$), we assign value i to the variable x_j. Let σ_π be the resulting variable assignment.

Given π, let

$$E_1 = \{e \mid e \in E \text{ and } \nexists j \in e\setminus\{\pi(e)\}, \exists i : \{\pi(e),j\} \subseteq R_i\},$$

and let $E_2 = E \setminus E_1$. In other words, E_1 is the set of hyperedges whose winner with π lies alone in some R_i, and E_2 is the set of hyperedges e whose winner with π lies in some R_i together with some other element(s) of e. We now bound the cardinality of E_2:

$$\begin{aligned} |E_2| &\leq \sum_{a=2}^{k}\sum_{i=1}^{t} \left(\binom{|R_i|}{a}\cdot\binom{n-|R_i|}{k-a}\right) \\ &\leq \sum_{a=2}^{k}\sum_{i=1}^{t} \frac{|R_i|^a\cdot(n-|R_i|)^{k-a}}{a!\cdot(k-a)!} \end{aligned}$$

$$\leq \sum_{a=2}^{k}\sum_{i=1}^{t}\frac{\left(\frac{n}{t}+1\right)^a\cdot n^{k-a}}{a!\cdot(k-a)!} \;\leq\; \sum_{a=2}^{k}\sum_{i=1}^{t} O\left(\epsilon^a\cdot n^k\right)$$
$$= \sum_{a=2}^{k} O\left(\epsilon^{a-1}\cdot n^k\right) \;\leq\; O\left(\epsilon\cdot n^k\right) \;=\; c'\cdot\epsilon\cdot n^k,$$

for some constant $c' > 0$. Now, for each $e \in E_1$, the constraint $C_{\pi(e),e}$ will be satisfied, and no other $C_{i_j,e}$, $i_j \in e \setminus \{\pi(e)\}$, will be satisfied. Thus, if $e \in E_1$, then the constraints in Γ_e will contribute $\tau - w_e(\pi(e))$ to the value of the k-CSP solution σ_π. Moreover, if $e \in E_2$, then the constraints in Γ_e will not contribute to the value of the k-CSP solution σ_π. Then, the value $v(\sigma_\pi)$ of the k-CSP solution σ_π can be lower bounded as

$$\begin{aligned} v(\sigma_\pi) &= \sum_{e\in E_1}(\tau - w_e(\pi(e))) \\ &\geq \sum_{e\in E_1}(\tau - w_e(\pi(e))) + \sum_{e\in E_2}(\tau - w_e(\pi(e))) - \tau|E_2| \\ &= \sum_{e\in E}(\tau - w_e(\pi(e))) - \tau\cdot|E_2| \\ &\geq |E|\cdot\tau - C(\pi) - c'\cdot\tau\cdot\epsilon\cdot n^k. \end{aligned}$$

Moreover, if σ is any assignment to the k-CSP variables, let π_σ be any permutation that ranks the elements decreasingly by assignment value (i.e., $i \prec_{\pi_\sigma} j$ if $x_i < x_j$) breaking ties arbitrarily. Let $E_1' = \{e \in E \mid \exists i \in e\, \forall j \in e \setminus \{i\} : x_i > x_j\}$. Then, if $e \in E_1'$, it holds that the hyperedge e contributes a value of $w_e(\pi_\sigma(e))$ to $C(\pi_\sigma)$. Then,

$$\begin{aligned} |E|\cdot\tau - C(\pi_\sigma) &= \sum_{e\in E}(\tau - w_e(\pi_\sigma(e))) \\ &\geq \sum_{e\in E_1'}(\tau - w_e(\pi_\sigma(e))) \;=\; v(\sigma). \end{aligned}$$

Then, $C(\pi_\sigma) \leq |E|\cdot\tau - v(\sigma)$.

Let $\pi^\star = \arg\min_\pi C(\pi)$ be a permutation that minimizes the WFHS cost. We have proved that $v\left(\sigma_{\pi^\star}\right) \geq |E|\cdot\tau - C(\pi^\star) - c'\cdot\tau\cdot\epsilon\cdot n^k$, i.e.,

$$C(\pi^\star) \geq |E|\cdot\tau - v(\sigma_{\pi^\star}) - c'\cdot\tau\cdot\epsilon\cdot n^k.$$

Now, let $\sigma^\star = \arg\max_\sigma v(\sigma)$. By $v(\sigma^\star) \geq v(\sigma_{\pi^\star})$, we have

$$C(\pi^\star) \geq |E|\cdot\tau - v(\sigma^\star) - c'\cdot\tau\cdot\epsilon\cdot n^k.$$

The algorithm in Theorem 5 of Schudy (2012), when run on the max-k-CSP instance $\mathcal{I}$ returns (in time $O(n^k)$) an assignment $\tilde\sigma$ such that $\mathrm{E}\left[v(\tilde\sigma)\right] \geq v(\sigma^\star) - \epsilon\cdot\tau\cdot n^k$. Then,

$$\begin{aligned} C(\pi^\star) &\geq |E|\cdot\tau - v(\sigma^\star) - c'\cdot\tau\cdot\epsilon\cdot n^k \\ &\geq |E|\cdot\tau - \mathrm{E}\left[v(\tilde\sigma)\right] - (c'+1)\cdot\tau\cdot\epsilon\cdot n^k. \end{aligned}$$

On the other hand, the permutation $\pi_{\tilde\sigma}$ will satisfy $\mathrm{E}\left[C(\pi_{\tilde\sigma})\right] \leq |E|\cdot\tau - \mathrm{E}\left[v(\tilde\sigma)\right]$. Thus,

$$C(\pi^\star) \geq \mathrm{E}\left[C(\pi_{\tilde\sigma})\right] - (c'+1)\cdot\tau\cdot\epsilon\cdot n^k,$$

or equivalently,

$$\mathrm{E}\left[C(\pi_{\tilde\sigma})\right] \leq C(\pi^\star) + (c'+1)\cdot\tau\cdot\epsilon\cdot n^k.$$

The algorithm then returns an expected additive $(c'+1)\cdot\tau\cdot\epsilon\cdot n^k$ approximation. Markov's inequality ensures that, if we run the algorithm $r = O(\alpha\log n)$ times, then with probability at least $1 - n^{-\alpha}$, the best of the r returned solutions is a $\left(c\cdot\tau\cdot\epsilon\cdot n^k\right)$-additive approximation for $c = 2(c'+1)$.

Finally, given that c is a constant and that ϵ can be chosen arbitrarily, we can substitute the value of ϵ with $\frac{\epsilon}{c}$ to guarantee that the running time will still be not larger than $O(\alpha n^k\log n)$ and that the algorithm returns a $(\tau\cdot\epsilon\cdot n^k)$-additive approximation with probability $1 - n^{-\alpha}$. □

In our application, we will have $\tau = \Theta(n^{-k})$, so that the additive error will be as small as ϵ for any constant $\epsilon > 0$.

5 RECONSTRUCTING RUMS FROM k-SLATES

Using Theorem 6, we give next an approximate separation oracle for $\overline{F}_\rho$ when $\mathcal{S}$ is the class of all slates of size k. Recall that $\overline{F}_\rho$ is defined by (3).

Theorem 7. *Let k be a constant and let $\mathcal{S} = \binom{[n]}{k}$. Fix any constants $\alpha > 0$ and $0 < \epsilon < \frac{1}{2}$. Then, there exists a randomized algorithm such that, given as input an assignment $\{D\} \cup \{\overline{\Delta}_{S,i}\}_{i\in S\in\mathcal{S}}$ to $\overline{F}_\rho$, in time $n^{O(k)}$ and with probability at least $1 - n^{-\alpha}$: (i) if at least one of the constraints c_ρ or $c_{S,i}$ (for $i \in S \in \mathcal{S}$) is unsatisfied, it returns an unsatisfied constraint of $\overline{F}_\rho$; or, (ii) if there exists at least one $\pi \in \mathbf{S}_n$ such that $\sum_{S\in\mathcal{S}} \overline{\Delta}_{S,\pi(S)} < D + 1 - 2\epsilon$, it returns an unsatisfied constraint of type c_π; otherwise, (iii) it might not return any unsatisfied constraint (even if some exists).*

Proof. First, we check the validity of every constraint of type c_ρ or $c_{S,i}$ (for $i \in S \in \mathcal{S}$); tshis can be done in time $O(n^{k+1})$. If any one of these constraints is unsatisfied, one of them is returned.

Otherwise, we run the algorithm of Theorem 6 on the WFHS instance given by the $\{\overline{\Delta}_{S,i}\}_{i\in S\in\mathcal{S}}$. This instance is (τ,k)-bounded with $\tau = 2|\mathcal{S}|^{-1}$. Thus, the algorithm of Theorem 6 returns, with probability at least $1 - n^{-\alpha}$, an $(\epsilon\cdot\tau\cdot|\mathcal{S}|)$-additive approximation, which is a (2ϵ)-additive approximation, in polynomial time. In particular, with that probability, it returns a permutation $\tilde\pi$ such that $\sum_{S\in\mathcal{S}} \overline{\Delta}_{S,\tilde\pi(S)} \leq \left(\min_\pi \sum_{S\in\mathcal{S}} \overline{\Delta}_{S,\pi(S)}\right) + 2\epsilon$.

The algorithm then checks the validity of the constraint $c_{\tilde\pi}$: if it is violated, it returns $c_{\tilde\pi}$ and none otherwise.

Observe that if $\min_\pi \sum_{S\in\mathcal{S}} \overline{\Delta}_{S,\pi(S)} < D + 1 - 2\epsilon$, the $\{\overline{\Delta}\}_{i\in S\in\mathcal{S}}$ assignment violates $c_{\tilde{\pi}}$, which will then be returned. Otherwise if $\min_\pi \sum_{S\in\mathcal{S}} \overline{\Delta}_{S,\pi(S)} \geq D + 1 - 2\epsilon$, the $c_{\tilde{\pi}}$ constraint might or might not be violated. □

Theorem 7 provides an approximate separation oracle for $\overline{F}_\rho$. If we plug it into the ellipsoid algorithm (Grötschel et al., 1988), we obtain a polynomial-time algorithm to compute a δ-additive approximation to our RUM fitting problem.

Theorem 8. *Let $k > 0$ be any integer and let δ be a constant in $0 < \delta < 1$. Then, Problem 3 can be approximated to within an additive value of δ in time $n^{O(k)}$, where n is the number of elements of the RUM.*

Proof. We describe RIPPLEK, a polynomial-time algorithm with the stated complexity that uses the ellipsoid algorithm of Grötschel et al. (1988) as a subroutine. As it is well-known, the ellipsoid algorithm uses a separation oracle as a black-box. In our case, the black-box is the approximate separation oracle of Theorem 7.

First, RIPPLEK guesses $\rho \in \left\{i \cdot \delta/2 \mid 0 \leq i \leq \lceil \frac{4}{\delta} \rceil\right\}$ (the algorithm performs a binary search among the values in this set). For a given ρ, the ellipsoid algorithm is invoked together with the approximate separation oracle of Theorem 7 to approximately check the non-emptiness of $\overline{F}_\rho$. In particular, the ellipsoid algorithm will call the separation oracle at most $n^{O(k)}$ many times[4] returning at most polynomially many separating hyperplanes. If such a set defines an infeasible LP, the ellipsoid algorithm correctly concludes that $\overline{F}_\rho$ is empty.

Otherwise, the ellipsoid algorithm returns a point $x = (D) \,.\, (\overline{\Delta}_{S,i})_{i\in S\in\mathcal{S}}$ that the oracle was unable to separate from $\overline{F}_\rho$. This point could lie inside of $\overline{F}_\rho$, or outside of it, since the oracle only guarantees that the constraints of type c_π hold to within an additive error of twice $\epsilon \triangleq \delta/4$. Let us define the point $x' = (D - \delta/2) \,.\, (\overline{\Delta}_{S,i})_{i\in S\in\mathcal{S}}$. We prove that $x' \in \overline{F}_{\rho-\delta/2}$. Indeed, each c_π constraint is satisfied by x' (c_π is off by at most $2\epsilon = \delta/2$ with the solution x, and the RHS of the c_π constraint decreases by $\delta/2$ when switching from x to x'). Moreover, the $c_{\rho-\delta/2}$ constraint of $\overline{F}_{\rho-\delta/2}$ is satisfied by x' (c_ρ is satisfied by x, thus $c_{\rho-\delta/2}$ is satisfied by x'). Each remaining constraint is also satisfied.

Let $i^\star$ be the largest i for which the algorithm establishes that $x' \in \overline{F}_{\rho^\star-\delta/2}$ where $\rho^\star = i^\star \cdot \delta/2$.[5] Then, the dual LP (2) does not admit a solution of value at least $\rho^\star + \delta/2$, but admits a solution of value at least $\rho^\star - \delta/2$. It follows that the optimal solution of the dual LP (2), and thus of the primal LP (1), lies in $[\rho^\star - \delta/2, \rho^\star + \delta/2]$. Hence, the ellipsoid algorithm with the above separation oracle, returns a solution that approximates the optimal solution of Problem 3 to within δ.

Finally, to recover an approximating RUM whose average-distance error is at most the smallest possible plus δ, RIPPLEK acts as follows. Consider the run of the ellipsoid algorithm with $\rho = \rho^\star$. In this run, the ellipsoid algorithm calls the separation oracle at most polynomially many times and returns no more than polynomially many separating hyperplanes. Some of these hyperplanes might refer to non-permutation constraints, and the rest refer to the permutation constraints of, say, permutations $\pi_1, \ldots, \pi_t$ (for $t \leq n^{O(k)}$). By restricting the primal LP (1) to its non-permutation variables and to the permutation variables $p_{\pi_1}, \ldots, p_{\pi_t}$, we obtain an LP of size $n^{O(k)}$ (i.e., solvable in time $n^{O(k)}$), and whose optimal value is at most δ plus the optimum of the primal LP (1). Thus, solving the restricted LP allows us to obtain a RUM with an error no greater than the smallest possible plus δ. □

Succinct Representation. As shown in Chierichetti et al. (2021), every RUM can be sketched to $O(\epsilon^{-2} \cdot k \cdot n \log^2 n)$ bits in such a way that the probability distribution of *each* slate of size at most k is approximated to within an ℓ_1-error of ϵ. This sketch is a RUM over $O(\epsilon^{-2} \cdot k \cdot \log n)$ permutations. Consequently, the approximately optimal RUM returned by the algorithm of Theorem 8 can be reduced (with the same $O(\delta)$ additive error with respect to the optimal approximating RUM) to a RUM supported by $O(\delta^{-2} \cdot k \cdot \log n)$ permutations.

Algorithm 1 A heuristic for Problem 3 RUMRUNNERK (Almanza et al., 2022).

$P \leftarrow \varnothing$
$\pi^\star \leftarrow$ any permutation from $\mathbf{S}_n$
repeat
  $P \leftarrow P \cup \{\pi^\star\}$
  Solve the primal LP (1) restricted to the variables $\epsilon_{S,i}$ for $i \in S \in \mathcal{S}$, and p_π for $\pi \in P$; let $\mathcal{P}$ be its optimal primal solution, and $\mathcal{D}$ be its optimal dual solution (i.e., the solution of LP (2))
  $\pi^\star \leftarrow$ `Viol-HP` $(\mathcal{D})$
until $\pi^\star = \perp$
return the RUM that samples $\pi \in P$ with probability $\mathcal{P}(p_\pi)$ and $\pi \in \mathbf{S}_n \setminus P$ with probability 0.

6 A NON-ADAPTIVE LOWER BOUND

Our reconstruction algorithm leverages on knowing the winning distribution of each slate of size k. We show here that a non-adaptive algorithm that aims to approximate the winning distribution of each slate of size k, must access a constant fraction of slates of size k during learning.

Theorem 9. *Let $\mathcal{A}$ be a non-adaptive algorithm that only queries an ϵ fraction of the slates of size $k \geq 2$. Then,*

[4] In our separation oracle, we set $\alpha = c \cdot k$ for some constant c; this guarantees that each call to the separation oracle will have the approximation properties of Theorem 7 with high probability.

[5] Such an $i^\star$ exists since the maximum ℓ_1-distance between two probability distributions is $2 \leq \lceil \frac{4}{\delta} \rceil \cdot \frac{\delta}{2}$.

Algorithm 2 A randomized local-search for `Viol-HP`. In experiments, we set $t = 100$ and $t' = 5$.

For $\pi \in \mathbf{S}_n$, let $\text{wfhs}(\pi) = \sum_{S \in \mathcal{S}} \mathcal{D}(\Delta_{S,\pi(S)})$
For $\pi \in \mathbf{S}_n$, let $N(\pi)$ be the set of permutations that can be obtained from π by moving one of its elements
let $0 < t' \leq t$ be two integers
$\text{wfhs}_{\min} \leftarrow \infty$
for $i = 1, \ldots, t$ **do**
  $\pi \leftarrow$ uniform at random permutation from $\mathbf{S}_n$
  while $\exists \pi' \in N(\pi)$ such that $\text{wfhs}(\pi') < \text{wfhs}(\pi)$ **do**
    $\pi \leftarrow \arg\min_{\pi' \in N(\pi)} \text{wfhs}(\pi')$
  if $\text{wfhs}(\pi) < \text{wfhs}_{\min}$ **then**
    $\text{wfhs}_{\min} \leftarrow \text{wfhs}(\pi)$
    $\pi_{\min} \leftarrow \pi$
  if $\text{wfhs}_{\min} < \mathcal{D}(D)$ **and** $i \geq t'$ **then**
    return $\pi_{\min}$
return $\perp$

with probability at least $1 - \epsilon$, the expected ℓ_1-error of $\mathcal{A}$'s prediction on at least one slate of size k is at least $2 - \frac{2}{k}$.

Proof. This result can be proved with a very simple RUM. Let S be a slate sampled uniformly at random from the class $\binom{[n]}{k}$, and let i be sampled uniformly at random from the slate S. The RUM $R = R^{i,S}$ will be supported by a single permutation $\pi_{i,S}$ that has the element of $[n] \setminus S$ in its first $n - |S|$ positions (sorted arbitrarily), element i in its $(n - |S| + 1)$st position, and the element of $S \setminus \{i\}$ in its last $|S| - 1$ positions (again, sorted arbitrarily).

Observe that $R_S(i) = 1$ and $R_S(j) = 0$, for each $j \in S \setminus \{i\}$. Moreover, if one queries R on any slate in $\binom{[n]}{k} \setminus \{S\}$, one is unable to tell which element of S ranks highest in π and, thus, in permutations sampled from R. Indeed, the elements of S lie in the last k positions of π and thus no slate of size k other than S will result in some element of S winning.

Thus, a non-adaptive algorithm that did not query S in its learning phase, can correctly guess that one element i' of S will always win in S. However, from the algorithm's perspective, $\Pr[j = i'] = \frac{1}{|S|}$ for each $j \in S$. Thus, if the goal of the algorithm is to minimize the expected ℓ_1-error on its guess for D_S, it should return the uniform vector $\left(\frac{1}{|S|}, \ldots, \frac{1}{|S|}\right)$, since it is the (geometric) median of the k possible distributions for D_S, and since each of these distributions is equally likely. The expected ℓ_1-error on D_S of any non-adaptive algorithm that does *not* query S during its learning phase is then at least $1 \cdot \left(1 - \frac{1}{|S|}\right) + (|S| - 1) \cdot \left(\frac{1}{|S|} - 0\right) = 2 - \frac{2}{|S|} = 2 - \frac{2}{k}$.

Moreover, if the non-adaptive algorithm queries only $\epsilon \cdot \binom{n}{k}$ slates during its learning phase, then it will query S with probability at most ϵ. If this event does not happen, the algorithm's expected ℓ_1-error on S is $\geq 2 \cdot \left(1 - \frac{1}{k}\right)$. □

Dataset	n	k	$\lvert P\rvert$	average error	lower bound	MNL avg. err.
`Sushi`	10	2	46	0	0	$2 \cdot 10^{-5}$
		3	241	0	0	0.0389
		4	631	0	0	0.0354
		5	916	0.0002	0	0.0282
`SFwork`	6	2	16	0	0	$2 \cdot 10^{-5}$
		3	36	0.0044	0.0044	0.0317
		4	35	0.0072	0.0072	0.0235
		5	20	0.0035	0.0035	$2 \cdot 10^{-5}$
`SFshop`	8	2	29	0	0	$2 \cdot 10^{-5}$
		3	109	0.0002	0.0002	0.0493
		4	195	0.0010	0.0010	0.0348
		5	192	0.0017	0.0017	0.0203
`A5`	16	2	121	0	0	$1 \cdot 10^{-5}$
		3	951	0.0005	0	0.0443
		4	1044	0.0105	0	0.0497
		5	1001	0.0207	0.0078	0.0542
`A9`	12	2	67	0	0	$1 \cdot 10^{-5}$
		3	441	0	0	0.0318
		4	1053	0.0014	0	0.0347
		5	1032	0.0080	0	0.0387
`A17`	13	2	79	0	0	$3 \cdot 10^{-5}$
		3	573	0	0	0.0441
		4	829	0.0127	0.0038	0.0603
		5	843	0.0281	0.0253	0.0732
`A48`	10	2	46	0	0	$1 \cdot 10^{-5}$
		3	241	0	0	0.0319
		4	561	0.0026	0.0002	0.0392
		5	433	0.0224	0.0208	0.0507
`A81`	11	2	56	0	0	$2 \cdot 10^{-5}$
		3	325	0.0005	0.0005	0.0484
		4	513	0.0250	0.0233	0.0659
		5	425	0.0535	0.0521	0.0859

Table 1: Results of the fitting experiments. $|P|$ is the size of the support found by RUMRUNNERK. For some datasets we were able to find a non-trivial lower bound on the average error achievable via a RUM. The last column represents the average ℓ_1-error of the MNL model.

7 EXPERIMENTS

We perform two types of experiments. In Section 7.2 we are given as input a set of slates and for each slate a winner, chosen by a user, over its elements. We evaluate the quality of our algorithm in representing the resulting winner distributions by measuring the ℓ_1-error between the winner distributions induced by the dataset and those given by the RUM found by the algorithm. Next, in Section 7.3, we consider a prediction setting where the data is split into training and test set. We learn a RUM using our algorithm on the training set and evaluate its generalization quality on the test set.

7.1 Experimental Setup

For practical reasons we did not implement the polynomial-time algorithm RIPPLEK, but based on similar ideas, used

the heuristic RUMRUNNERK described in Algorithm 1, which is from Almanza et al. (2022). The heuristic needs access to a separation oracle (`Viol-HP`) that, as shown in Section 5, can be seen as an instance of the WFHS problem. We implemented such an oracle both with an exact algorithm running in time $O(n^k 2^n)$ (which we describe in the Appendix, and that is a generalized version of the exact algorithm of Lawler (1964) for the FAS problem) and also using a randomized local-search heuristic described in Algorithm 2; the latter is a generalization of the algorithm in Almanza et al. (2022). We were able to use the exact oracle on most of the datasets, but when it turned out to be too slow, we resorted to the local-search (in particular, we needed the local-search on dataset `A5`). Note that, when provided with an exact oracle, RUMRUNNERK is guaranteed to find the optimal RUM, however, we have no (non-trivial) upper bound on its running time. On the other hand, when we use the local-search heuristic, RUMRUNNERK loses also its guarantee to converge to the optimal RUM, since the local-search might fail at finding a separating hyperplane even if it exists. We implemented RUMRUNNERK in Python using IBM cplex[6] and we ran it on general-purpose hardware[7].

Baselines. We compared the performances of our algorithm with the MNL model (Bradley and Terry, 1952), both in the fitting experiments and in the prediction ones. We used the scikit-learn[8] implementation of this model.

Slate sizes. We ran the experiments on sets of equal-sized slates, in particular, we considered slates of size between 2 and 5, i.e., $\mathcal{S} \subseteq \binom{[n]}{k}$ for $k \in \{2, 3, 4, 5\}$. Note that we allowed missing k-slates (i.e., possibly $\mathcal{S} \subsetneq \binom{[n]}{k}$).

Datasets. We performed the experiments on the following:

(i) `Sushi` dataset (Kamishima, 2003) contains a list of permutations over 10 elements, representing people's preferences over different types of sushi. Since we only care about k-slates, we transformed each permutation into $\binom{n}{k}$ different k-slates setting the winner according to the permutation. Note that a RUM with error 0 exists on such a dataset.

(ii) Datasets `SFwork` and `SFshop` (Koppelman and Bhat, 2006) contain a list of choices between different transportation alternatives made by people going to work or to a shopping center, respectively. Since these datasets contain only a few slates of fixed size, we augmented them in the following way: each slate S, $|S| \geq k$, with winner $w \in S$, is transformed into $\binom{|S|-1}{k-1}$ slates of size k, each having w as a winner. This transformation might induce a bias, however, it seems reasonable in practice.

(iii) Datasets `A5`, `A9`, `A17`, `A48`, `A81` (Tideman, 2006) contain lists of election ballots, which are partial ordering of the elements (i.e., each ballot is a sorted subset of S). As in `Sushi`, we converted the sorted subsets into several k-slates assigning the winner according to the ordering.

Dataset	n	k	RUMRUNNERK	MNL	Train Tensor
`Sushi`	10	2	0.023	0.023	0.023
		3	0.027	0.037	0.027
		4	0.028	0.033	0.029
		5	0.028	0.030	0.030
`SFwork`	6	2	0.091	0.088	0.088
		3	0.094	0.087	0.094
		4	0.085	0.074	0.081
		5	0.071	0.072	0.072
`SFshop`	8	2	0.081	0.080	0.081
		3	0.066	0.067	0.066
		4	0.060	0.057	0.062
		5	0.056	0.051	0.058
`A9`	12	2	0.046	0.046	0.046
		3	0.058	0.055	0.059
		4	0.065	0.062	0.071
		5	0.070	0.069	0.081
`A17`	13	2	0.107	0.106	0.107
		3	0.141	0.128	0.147
		4	0.156	0.150	0.177
		5	0.170	0.168	0.201
`A48`	10	2	0.071	0.071	0.071
		3	0.094	0.084	0.094
		4	0.105	0.097	0.114
		5	0.117	0.112	0.132
`A81`	11	2	0.091	0.090	0.091
		3	0.121	0.113	0.126
		4	0.143	0.138	0.158
		5	0.168	0.166	0.193

Table 2: Results of the prediction experiments. We run a 5-fold cross validation with 10 different seeds for each dataset and algorithm. The table reports the (avg.) RMSE of each algorithm. The standard deviations are in $[0.013, 0.025]$ for `SFwork` and in $[0.001, 0.009]$ for others.

7.2 RUM Fitting

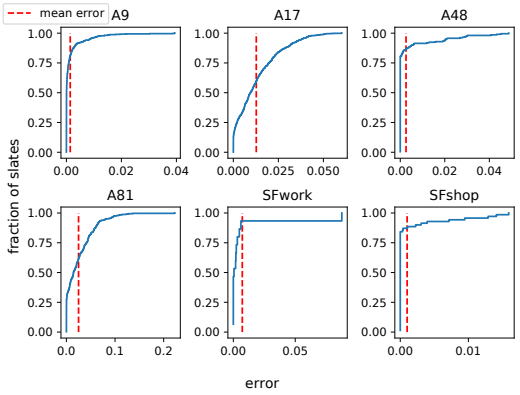
The results of the fitting experiments are shown in Table 1. We let RUMRUNNERK run for a maximum of 1500 iterations and stop it earlier if the average error decreased by less than 10^{-5} in 20 iterations (meaning that the algorithm converged). Note that our heuristic always obtains a smaller error than the MNL model, and it seems therefore more suitable for representing the datasets.

Due to the use of the local-search heuristic and to the fact that we stop the algorithm after a fixed number of iterations, RUMRUNNERK will not, in general, converge to the optimal RUM. However, it is possible to find lower bounds on the best possible average error achievable via a RUM. Consider the dual LP (2) restricted to have only the permutation constraints of P, $P \subseteq \mathbf{S}_n$. Consider an opti-

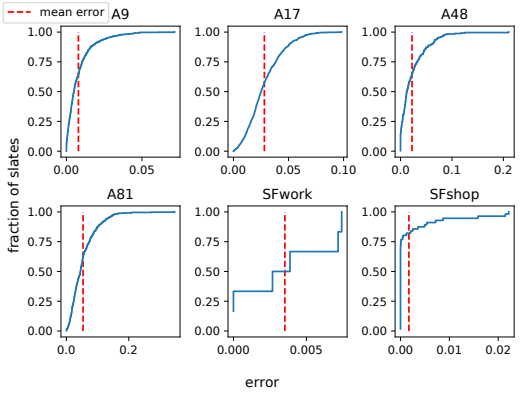
[6] `https://www.ibm.com/analytics/cplex-optimizer`

[7] Intel core i7, 8GB of RAM

[8] `https://scikit-learn.org/stable/`



(a) $k = 4$



(b) $k = 5$

Figure 1: Distribution of the errors over the slates. Fixed an error x, the corresponding y value is the ratio $\frac{|\{S|S\in\mathcal{S} \text{ and } |R_S - D_S|_1 \leq x\}|}{|\mathcal{S}|}$, where R is the computed RUM and D_S is the empirical distribution over S.

mal solution $\mathcal{D}$ to such restricted LP having value v and suppose we know, via an exact algorithm for WFHS, that $y = \min_{\pi\in\mathbf{S}_n} \sum_{S\in\mathcal{S}} \mathcal{D}(\Delta_{S,\pi(S)})$. If $y \geq \mathcal{D}(D)$, then v is the optimal value of the original dual LP (2), and therefore of the primal LP (1). Otherwise, note that the point $(y) \cdot (\Delta_{S,i})_{i\in S\in\mathcal{S}}$ is a feasible solution to the dual LP (2) and it has value $x = v - (\mathcal{D}(D) - y)$, therefore x is a lower bound on the optimal value of the primal LP (1). Fortunately, this trick to find a lower bound can be done at the end of the fitting procedure, so to run the exact algorithm for WFHS only once. Using these simple remarks we were able to find non-trivial lower bounds on the average error for several of the considered datasets. In practice, RUMRUNNERK often finds the optimal (or almost-optimal) RUM; furthermore, the error of RUMRUNNERK seems to increase as we increase the size of the slates. We analyzed how the errors are distributed (see Figure 1) and it seems that the majority of the slates incur an error below the average, in particular, note that most of the 4-slates have an error close to 0.

7.3 Quality of Predictions

We performed an ℓ-fold cross-validation to assess the generalization quality of RUMRUNNERK. In particular, given the slates (with repetitions), we divided them into ℓ groups of roughly the same size, built at random. We then performed ℓ iterations: during the ith iteration, the ith group is used as the test set and the union of the other $\ell - 1$ groups is used as the training set. In our experiments, we set $\ell = 5$ and we repeated the random splitting 10 times with different seeds.

As in Almanza et al. (2022); Makhijani and Ugander (2019), since the splitting is performed on the initial slates, the same slate might appear both in training and testing. Following them, we used root mean-squared error $\text{RMSE}(D, \hat{D}) = \sqrt{|\mathcal{S}|^{-1} \sum_{i\in S\in\mathcal{S}} (D_S(i) - \hat{D}_S(i))^2}$ to evaluate the performances of the algorithms. We trained RUMRUNNERK for a maximum of 250 iterations. The results are shown in Table 2. (The results for the case $k = 2$ differ from Almanza et al. (2022) because they normalized by $\binom{n}{2}$ rather than $|\mathcal{S}|$, however, using their definition we get essentially the same results.) Note that the algorithm gives performances comparable to the MNL model; furthermore, using the training data directly to make predictions[9] gives results only slightly worse than RUMRUNNERK, this is due to the fact that RUMRUNNERK can represent the input data with a very small error.

8 CONCLUSIONS

In this paper we obtained a polynomial-time algorithm for finding a RUM that best approximates a given set of winning distributions for slates of any constant size k. While our lower bound shows that the reconstruction algorithm is query-optimal, extending it work for adaptive algorithms is an interesting research direction. Developing provably good algorithms that avoid the ellipsoid method is also an intriguing question that merits further investigation.

Acknowledgments

We thank Pasin Manurangsi for useful discussions and suggestions. Flavio Chierichetti and Alessandro Panconesi were supported in part by BiCi—Bertinoro International Center for Informatics. Flavio Chierichetti was supported in part by the Google Gift "Algorithmic and Learning Problems in Discrete Choice" and by the PRIN project 2017K7XPAN.

References

Matteo Almanza, Flavio Chierichetti, Ravi Kumar, Alessandro Panconesi, and Andrew Tomkins. RUMs

[9] We used the uniform distribution for slates that appear only in the test set.

from head-to-head contests. In *ICML*, pages 452–467, 2022.

K Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2003.

Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Learning a mixture of two multinomial logits. In *ICML*, pages 961–969, 2018a.

Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Light RUMs. In *ICML*, pages 1888–1897, 2021.

Nir Rosenfeld, Kojin Oshiba, and Yaron Singer. Predicting choice with set-dependent aggregation. In *ICML*, pages 8220–8229, 2020.

A. Seshadri, S. Ragain, and J. Ugander. Learning rich rankings. In *NeurIPS*, 2020.

Arjun Seshadri, Alex Peysakhovich, and Johan Ugander. Discovering context effects from raw choice data. In *ICML*, pages 5660–5669, 2019.

Daniel McFadden and Kenneth Train. Mixed MNL models for discrete response. *J. Applied Econometrics*, 15(5): 447–470, 2000.

Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Discrete choice, permutations, and reconstruction. In *SODA*, pages 576–586, 2018b.

Vivek F. Farias, Srikanth Jagabathula, and Devavrat Shah. A data-driven approach to modeling choice. In *NIPS*, pages 504–512, 2009.

Hossein Azari Soufiani, David C. Parkes, and Lirong Xia. Random utility theory for social choice. In *NIPS*, pages 126–134, 2012.

Sewoong Oh and Devavrat Shah. Learning mixed multinomial logit model from ordinal data. In *NIPS*, pages 595–603, 2014.

Sahand Negahban, Sewoong Oh, Kiran K. Thekumparampil, and Jiaming Xu. Learning from comparisons and choices. *JMLR*, 19(1):1478–1572, 2018.

Wenpin Tang. Learning an arbitrary mixture of two multinomial logits. *arXiv*, 2007.00204, 2020.

Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *STOC*, pages 95–103, 2007.

Warren Schudy. *Approximation Schemes for Inferring Rankings and Clusterings from Pairwise Data*. PhD thesis, Brown University, 2012.

Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

Grigory Yaroslavtsev. Going for speed: Sublinear algorithms for dense r-CSPs. *arXiv*, 1407.7887, 2014. URL `http://arxiv.org/abs/1407.7887`.

Yuichi Yoshida and Yuan Zhou. Approximation schemes via Sherali–Adams hierarchy for dense constraint satisfaction problems and assignment problems. In *ITCS*, page 423–438, 2014. See also `https://yuanz.web.illinois.edu/papers/ptas-by-hierarchy.pdf`.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer, 1988.

E. Lawler. A comment on minimum feedback arc sets. *IEEE Transactions on Circuit Theory*, 11(2):296–297, 1964.

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.

Toshihiro Kamishima. Nantonac collaborative filtering: Recommendation based on order responses. In *KDD*, pages 583–588, 2003. doi: 10.1145/956750.956823. URL `https://doi.org/10.1145/956750.956823`.

Frank Koppelman and Chandra Bhat. *A Self Instructing Course in Mode Choice Modelling: Multinomial and Nested Logit Models*. U.S. Department of Transportation, Federal Transit Administration, 2006.

Nicolaus Tideman. *Collective Decisions and Voting: The Potential for Public Choice*. Routledge, 2006.

Rahul Makhijani and Johan Ugander. Parametric models for intransitivity in pairwise rankings. In *WWW*, pages 3056–3062, 2019.

A AN EXACT ALGORITHM FOR WFHS

In this section we give Algorithm 3, a dynamic programming for the WFHS problem (Problem 4). Our Algorithm generalizes the algorithm from Lawler (1964) for the FAS problem.

Algorithm 3 An Algorithm for WFHS.

$C(\varnothing) \leftarrow 0$
for $j = 1, \ldots, n$ **do**
  for $A \in \binom{[n]}{j}$ **do**
    for $a \in A$ **do**
      $t_a \leftarrow 0$
    for $e \in E$ such that $e \subseteq A$ **do**
      for $a \in e$ **do**
        $t_a \leftarrow t_a + w_e(a)$
    $C(A) \leftarrow \min_{a \in A} (t_a + C(A \setminus \{a\}))$
    $\ell(A) \leftarrow a$, for any a such that $t_a + C(A \setminus \{a\}) = C(A)$
$A_n \leftarrow [n]$
Let $\pi^\star$ be an array of size n
for $j = n, \ldots, 1$ **do**
  $\pi^\star_j \leftarrow \ell(A_j)$
  $A_{j-1} \leftarrow A_j \setminus \{\ell(A_j)\}$
return $\pi^\star$

Theorem 10. *Algorithm 3 returns an optimal solution to WFHS in time* $O\left(k \cdot |E| \cdot 2^n\right) \leq O\left(n^k \cdot 2^n\right)$.

Proof. For a given $A \subseteq [n]$, let $C(A)$ be the minimum WFHS cost of a solution to the instance projected on the elements of A (i.e., to the instance obtained by removing each hyperedge that is not fully contained in A). Then, $C(\varnothing) = 0$ and

$$C(A) = \min_{a \in A} \left(C(A \setminus \{a\}) + \sum_{\substack{e \in E \\ a \in e \subseteq A}} w_e(a) \right).$$

Then, the time required to fill entry A of the array is at most $O\left(|E| \cdot k\right) = O(n^k)$ — indeed, we can initialize one variable $t_a = 0$ for each $a \in A$, for a total of $|A| \leq n$ variables). We then iterate over the edges of E: for each $a \in e \subseteq A$, we add $w_e(a)$ to t_a. The value of $C(A)$ is then the minimum, over $a \in A$, of $C(A \setminus \{a\}) + t_a$; we also set $\ell(A)$ to be the item of A that achieves this minimum

Once array A is filled, a permutation $\pi^\star$ having minimum WFHS cost, $C(\pi^\star) = C([n])$, can be easily obtained. Let $A_n = [n]$. In general, the element in position $1 \leq j \leq n$ of $\pi^\star$ will be equal to $\ell(A_j)$, and for $1 \leq j \leq n$, $A_{j-1} = A_j \setminus \{\ell(A_j)\}$. Thus, an extra iteration over the positions $n, n-1, \ldots, 1$ is sufficient to obtain $\pi^\star$.

Finally, filling the array takes time $O(k \cdot |E| \cdot 2^n) \leq O(k \cdot \binom{n}{k} \cdot 2^n) \leq O(n^k \cdot 2^n)$ and computing $\pi^\star$ takes time $O(n)$. □