RUMs from Head-to-Head Contests

Matteo Almanza [1 2] **Flavio Chierichetti** [3] **Ravi Kumar** [4] **Alessandro Panconesi** [3] **Andrew Tomkins** [4]

Abstract

Random utility models (RUMs) encode the likelihood that a particular item will be selected from a slate of competing items. RUMs are well-studied objects in both discrete choice theory and, more recently, in the machine learning community, as they encode a fairly broad notion of rational user behavior. In this paper, we focus on slates of size two representing head-to-head contests. Given a tournament matrix M such that $M_{i,j}$ is the probability that item j will be selected from $\{i, j\}$, we consider the problem of finding the RUM that most closely reproduces M. For this problem we obtain a polynomial-time algorithm returning a RUM that approximately minimizes the average error over the pairs. Our experiments show that RUMs can *perfectly* represent many of the tournament matrices that have been considered in the literature; in fact, the maximum average error induced by RUMs on the matrices we considered is negligible (≈ 0.001). We also show that RUMs are competitive, on prediction tasks, with previous approaches.

1. Introduction

Random utility models (RUMs) are perhaps the most important model in discrete choice (Train, 2003). In full generality, they posit that a user selecting an item from some slate of choices has a personal vector encoding the value of all items in the universe. The user behaves rationally by selecting the available item in the slate of the highest value. As each user may have a different value vector, the RUM model allows a wide range of effects. There are no known efficient learning algorithms for RUMs.

In pairwise choice, the setting is the same, but the user must choose between only two alternatives, rather than from an arbitrary slate of alternatives as in the full model. Pairwise choice arises whenever head-to-head competitions occur, as in two-player games, two-alternative forced-choice testing (Bogacz et al., 2006), or online experiences that compare an item with an alternative. The pairwise choice model is well studied, with a number of approaches that learn over either subsets of the class of RUMs, or incomparable models (Chen & Joachims, 2016; Adams et al., 2010; Makhijani & Ugander, 2019; Veerathu & Rajkumar, 2021).

Our main technical findings are general RUM learning algorithms for pairwise choice. First, we introduce RIPPLE, for *Randomized Interior Point Permutation LEarner*. We show that for pairwise choice, RIPPLE is guaranteed to return a RUM that attains almost the best possible average error over all RUMs, in polynomial time. The polynomial-time guarantee requires solving an exponential-size dual linear program using the ellipsoid method, which is unlikely to be efficient in practice. Hence, we also introduce RUMRUNNER, a practical variant which is also guaranteed to return a near-optimal RUM. RUMRUNNER uses a separating hyperplane heuristic that performs efficiently in our experimental evaluation, but unlike RIPPLE it does not have a guarantee of polynomial running time. While previous results in this area require heuristics to optimize non-concave likelihoods, often requiring multiple runs to escape possible poor local optima, RUMRUNNER guarantees convergence to a solution whose value is ϵ-close to the optimum, and contains no hyper-parameters to tune. In empirical evaluations we observe that RUMRUNNER is neither subject to poor local optima nor has stability issues.

Our approach. A RUM may be characterized as a distribution over users, each of whom is represented by a vector giving the value of each item in the universe. Given a pair of alternatives, the likelihood that the first element is selected is the likelihood that a value vector drawn from the distribution assigns a higher score to the first item than the second. Since the decision depends only on which score is higher, we may represent these value vectors as permutations of $\{1, \ldots, n\}$ for a universe of n items, which simplifies the RUM to a distribution over permutations.

Our approach begins by writing a large linear program (LP) in which a variable encodes the weight of each permutation in the RUM. The LP contains constraints for each pair $\{i, j\}$ of elements asking that the total probability assigned

[1]Algorand Labs [2]This work was carried out while the author was at Sapienza University of Rome. [3]Sapienza University of Rome [4]Google Mountain View. Correspondence to: Matteo Almanza <almanza@di.uniroma1.it>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

to permutations with $i \succ j$ should have at most a given error from the true target probability. The objective of the LP then is to minimize the total approximation error.

This LP has exponentially many variables, but its dual has only polynomially many variables (albeit with exponentially many constraints). The structure of the dual allows a separating oracle to be constructed by approximately solving an instance of the minimum Feedback Arc Set (FAS) problem. This in turn gives a polynomial-size set of permutations capable of additively approximating the best possible RUM, and the original LP can now be solved in polynomial time to recover the approximating solution.

Empirical findings. We present a comprehensive set of experiments comparing the RUM algorithm to other standard and recent approaches to pairwise choice: the Blade-Chest (BC) algorithm of Chen & Joachims (2016) and the MV algorithm of Makhijani & Ugander (2019).

In the pairwise setting, the entire data distribution has a particularly simple parameterization: it can be completely represented by an $n \times n$ *tournament matrix* M such that $M_{i,j}$ is the probability that j beats i when the user is shown the pair $\{i, j\}$. Furthermore, for several of our experimental datasets, the training data provides a high-accuracy estimate of M. In these cases, the training data is effectively the entire data distribution, so the best algorithm will exactly reproduce the matrix M. Thus, in addition to comparing test-set prediction accuracy (which will depend on the variance of the entries of M), we also report the ability of an algorithm to capture the matrix M as given. For datasets based on election data, our RUM-finding algorithm is able to *perfectly* encode M. For other datasets, the error we attain in approximating M is on the order of $1/n^2$. For predictive tasks, our RUM algorithms also perform comparably to the BC and MV algorithms.

Implications. Finding a high-quality RUM for pairwise comparisons has several desirable properties:

- RUMs naturally provide the ability to predict winning probabilities for larger slates, even when trained only on pairwise data; no other algorithm we are aware of for this problem has this property.
- Our learned RUMs are based on weighted sets of permutations, and are thus naturally interpretable and introspectible. For instance, one can study the properties of the high-weight permutations.
- RUMs capture a natural notion of rational behavior, in which users select the item they value most highly.
- The permutations learned by our RUM algorithm represent a *de facto* clustering of the user population into segments, characterized by their different behaviors with respect to their choices. This may have value beyond predictions, e.g., in market segmentation.

The rest of the paper proceeds as follows. Section 2 covers related work and Section 3 contains the background. Section 4 presents our results for minimizing average errors and shows evidence that our approach will not extend to the uniform error setting. Section 5 describes techniques to find succinct approximating RUMs. Experimental results are in Section 6 and Section 7 has concluding thoughts. All missing proofs are in the Supplementary Material.

2. Related work

The problem of representing a tournament matrix with choice models has been often considered in the literature. We mention those that are closest in spirit to our work.

Chen & Joachims (2016) proposed the *Blade-Chest* (BC) model, which represents each player i with two vectors b_i (blade) and c_i (chest) in the Euclidean space; the probability that player i beats player j is a function of b_j, c_i, b_i, c_j. The BC model can produce any tournament matrix, in particular those with *non-transitive* triplets[1], that are observed in practice and cannot be achieved with standard MNL models. The BC model is optimized via gradient descent (GD) on a non-concave function, with no global optimum guarantee. Earlier, Adams et al. (2010) proposed a similar model to represent the outcomes of baseball matches.

Makhijani & Ugander (2019) proposed a different model, the *Majority Vote* (MV) model. In this model, each player is represented by k features, with k odd. Each feature corresponds to an *independent* RUM[2], e.g., an MNL or a Gaussian–Thurstone model. A match between two players i and j consists of k rounds: in the tth round, the two players play independently against each other using only their tth features. The player who wins the majority of rounds, is the winner of the match. They proposed GD to maximize the model's (non-concave) likelihood, with no guarantees on the global optimum. Interestingly, they show that for a class of models, the ability of models from the class to induce non-transitivity implies the non-log-concavity of their likelihoods, i.e., the GD procedure of any of these models could still get stuck at a local optimum.

In fact, Makhijani & Ugander (2019) suggest general (i.e., possibly dependent) RUMs to be a natural choice for representing tournament matrices, but point out that previous work (Train, 2003) has found optimization over unrestricted RUMs to be hard. In our paper we tackle this problem. We show how to find an approximately optimal unrestricted RUM for a tournament matrix; we circumvent the aforementioned non-concavity issues by casting the prob-

[1] Players i, j, k such that each beats another with probability more than $1/2$: $P_{i,j} > 1/2$, $P_{j,k} > 1/2$, and $P_{k,i} > 1/2$.

[2] An *independent RUM* is a RUM where the noise vector η (see Section 3.1) has mutually independent entries.

lem combinatorially, and solving it using LP and the ellipsoid method. We also observe experimentally that these unrestricted RUMs almost perfectly represent the tournament matrices that originated them—vindicating that RUMs are ideal for this task!

Finally, we mention the novel two-level (2L) model of Veerathu & Rajkumar (2021). In general their model is as powerful (and as hard to optimize) as the BC model. They thus restrict their model to rank-2 tournaments and show a polynomial time learning algorithm in this case.[3] Interestingly, their learning algorithm appeals to the minimum FAS problem, which they show can be solved optimally in polynomial time on rank-2 tournaments. We observe that there exist dependent RUMs that cannot be represented by the 2L model (see Appendix D).

The problem of actively, or passively, learning RUMs has been studied in several papers (Soufiani et al., 2012; Oh & Shah, 2014; Chierichetti et al., 2018a;b; Negahban et al., 2018; Tang, 2020). Farias et al. (2009); Chierichetti et al. (2021) considered the problem of bounding a RUM's support without changing the RUM's behavior. The goal of our paper, instead, is to find the RUM (approximately) closest to a given tournament matrix.

3. Preliminaries

3.1. Discrete Choice and Random Utility Models

In our paper, we use $[n]$ to denote the set $\{1, \ldots, n\}$, and $2^{[n]}$ to denote the power set of $[n]$. We also use $\binom{[n]}{k}$ to denote the class of subsets of $[n]$ of cardinality k.

We use $\mathbf{S}_n$ to denote the set of all the permutations of $[n]$, and $\mathbf{S}_n^\star = \mathbf{S}_n \setminus \{(1 \succ \cdots \succ n)\}$ to denote the set of all the permutations of $[n]$ with the exception of the permutation that reverses the natural ordering relation. For a permutation $\pi \in \mathbf{S}_n$ and for an item $i \in [n]$, we let $\pi(i) \in [n]$ be the value (or position) of item i in π. For instance, if $\pi = (3 \prec 1 \prec 4 \prec 2)$, then $\pi(2) = 4, \pi(4) = 3, \pi(1) = 2, \pi(3) = 1$.

As is standard in discrete choice literature, we use *slate* to denote a set in $2^{[n]} \setminus \{\varnothing\}$, i.e., any non-empty subset of $[n]$. For a permutation $\pi \in \mathbf{S}_n$ and a slate $T \subseteq [n]$, we let

$$\pi(T) = \arg\max_{i \in T} \pi(i),$$

denote the item of T that has the largest value in π, i.e., the *winner* in T given π.

A *random utility model (RUM)* on $[n]$ is a distribution D on $\mathbf{S}_n$. (Henceforth, we drop "$[n]$" when it is clear from the context.) For a slate T, we use D_T to denote the distribution of the random variable $\pi(T)$ for $\pi \sim D$. (Clearly, $\mathrm{supp}(D_T) \subseteq T$.) In other words, D_T is the induced distribution of the winner in T for a random permutation from D. Let $D_T(i)$ be the probability that i is a winner in T.

RUMs are often presented in terms of noisy item evaluations made by users: each item i has a base value V_i, each user samples $(\eta_1, \ldots, \eta_n)$ from a joint noise distribution[4], and the utility U_i of item i to this user is $V_i + \eta_i$. Given a slate, the "rational" user chooses the highest utility item in the slate (with ties broken u.a.r.). Equivalently, the user first sorts all the items decreasingly according to the observed U_i's (again, with ties broken u.a.r.) to get a permutation. Then, for a given slate, the rational user chooses the highest ranked item in the slate according to the permutation. It is easy to see that these two definitions are equivalent (Chierichetti et al., 2018a).

3.2. Tournaments and Feedback Arc Set problems

Given a set $[n]$ of players, and $\{i, j\} \in \binom{[n]}{2}$, a *tournament matrix* $\{P_{i,j}\}_{i,j=1}^n$ gives the (empirical) probability that j wins in a head-to-head contest with i. Clearly, $P_{i,j}, P_{j,i} \geq 0$ and $P_{i,j} + P_{j,i} = 1$, for each $\{i, j\} \in \binom{[n]}{2}$.

As we will show, RUM optimization for head-to-head contests (i.e., tournament matrices) is closely related to the *minimum Feedback Arc Set (FAS)* problem, which is NP-hard (Karp, 1972); the latter has two equivalent definitions.

The first definition is perhaps the most well-known. A *τ-bounded directed graph* $G(V, A, w)$ consists of the vertex set $V = [n]$, the arc set $A \subseteq \{(i, j) \in V^2 \mid i \neq j\}$ satisfying $(i, j) \in A \implies (j, i) \notin A$, and a *non-negative* weight function $w : A \to [0, \tau]$ on its arcs.

Problem 1 (FAS on τ-bounded directed graphs)**.** *Given a τ-bounded directed graph $G(V, A, w)$, find a permutation π of the vertices V that minimizes*

$$C'_{G(V,A,w)}(\pi) = \sum_{(i,j) \in A \text{ and } i \prec_\pi j} w((i, j)).$$

The second definition forces input graph to be a transitive tournament (i.e., a complete DAG) with arcs going from lower-indexed vertices to higher-indexed ones. A *τ-bounded transitive tournament* $G(V, A, w)$ is a complete DAG consisting of the vertex set $V = [n]$, arc set $A = \{(i, j) \in V^2 \mid i < j\}$, and a weight function $w : A \to [-\tau, \tau]$ on its arcs.

Problem 2 (FAS on τ-bounded transitive tournaments)**.** *Given a τ-bounded transitive tournament $G(V, A, w)$, find a permutation π of the vertices V that minimizes*

[3] In their experiments, they do not measure how close their guesses of the win-probabilities are; they consider the "upset" metric, which counts the number of pairs of players such that the model correctly identifies the most-likely winner in the pair.

[4] We recall that, in an independent RUM, the η_i's are mutually independent.

$$C''_{G(V,A,w)}(\pi) = \sum_{1\le i<j\le n \text{ and } i\prec_\pi j} w((i,j)).$$

Note that while the input graphs to Problem 2 are less general than those to Problem 1, the arc weights of Problem 2 can be chosen in a more general way.

For Problem 1, Frieze & Kannan (1999) give an $O(\delta\tau n^2)$-additive approximation algorithm, for $\delta > 0$.

Theorem 3 (Frieze & Kannan (1999))**.** *There exists an approximation algorithm for Problem 1 that, for a τ-bounded directed graph G on n nodes, returns a permutation π' such that $C'_G(\pi') \le O(\delta\tau n^2) + \min_{\pi\in\mathbf{S}_n} C'_G(\pi)$, in time $O\left(\delta^{-4}n\right) + 2^{\widetilde{O}(\delta^{-2})}$, with probability at least $2/3$, for an arbitrary $\delta > 0$.*

We will use this approximation algorithm for Problem 2. The two problems are equivalent from an additive approximation point of view. The following is folklore.

Observation 4. *Any α-additive approximation polynomial time algorithm for Problem 1 with a given τ can be transformed into an α-additive approximation polynomial time algorithm for Problem 2 with the same τ, and vice versa.*

3.3. Approximating Tournaments by RUMs

We formally define the main problems that we consider.

Definition 5 (Average RUM approximation)**.** *A given tournament matrix P is* approximated on average to within ϵ *by a RUM R if*

$$\operatorname*{avg}_{1\le i<j\le n} \left|R_{\{i,j\}}(j) - P_{i,j}\right| \le \epsilon.$$

Given P, let $\epsilon_1(P)$ be the smallest[5] *$x \ge 0$ such that there exists a RUM that approximates P on average to within x.*

Note that each RUM R on $[n]$ is such that $R_{\{i,j\}}(i) = 1 - R_{\{i,j\}}(j)$ for each $\{i,j\} \in \binom{[n]}{2}$. Thus, the above average is equal to $\frac{\sum_{i\in[n]}\sum_{j\in[n]\setminus\{i\}} |R_{\{i,j\}}(j) - P_{i,j}|}{n\cdot(n-1)}$, i.e., average the error over each pair and each winner of the pair.

Problem 6. *Given a tournament matrix P, find a δ-additive approximation to $\epsilon_1(P)$, i.e., obtain a RUM whose average distance from P is not larger than $\epsilon_1(P) + \delta$.*

Similarly, one can define the maximum-error variant.

Definition 7 (Uniform RUM approximation)**.** *A given tournament matrix P is* uniformly approximated to within ϵ *by a RUM R if*

$$\max_{1\le i<j\le n} \left|R_{\{i,j\}}(j) - P_{i,j}\right| \le \epsilon.$$

Given P, let $\epsilon_\infty(P)$ be the smallest[5] *$x \ge 0$ such that there exists a RUM that uniformly approximates P to within x.*

[5]As we will see, the minimum exists since it can be obtained as the solution to a finite-sized LP.

4. Minimizing Average Error

In this section we describe the RIPPLE algorithm to find a RUM that additively approximates $\epsilon_1(P)$ to within $\delta = \Theta\left(\log^{-0.49} n\right)$ in time polynomial in n. Our approach has three steps: (i) writing down Problem 6 as a linear program (LP) of exponential size, (ii) developing an approximate separation oracle for its dual, and (iii) showing that the Ellipsoid algorithm converges to an approximate solution when aided by this approximate separation oracle.

We start by writing down the primal LP for Problem 6:

$$\begin{cases} \min \binom{n}{2}^{-1} \cdot \sum_{1\le i<j\le n} \epsilon_{i,j} & \\ P_{i,j} - \epsilon_{i,j} \le \sum_{\substack{\pi\in\mathbf{S}^\star_n \\ i\prec_\pi j}} p_\pi \le P_{i,j} + \epsilon_{i,j} & \forall 1 \le i < j \le n \\ \sum_{\pi\in\mathbf{S}^\star_n} p_\pi \le 1 & \\ \epsilon_{i,j} \ge 0 & \forall 1 \le i < j \le n \\ p_\pi \ge 0 & \forall \pi \in \mathbf{S}^\star_n \end{cases} \tag{1}$$

The program minimizes the average ℓ_1-error over the pairs, optimizing over the permutations. Note that, since its number of variables is exponential in n, this LP cannot be directly solved as is in time polynomial in n.

Observe that the LP (1) assigns a non-negative probability to each permutation $\pi \in \mathbf{S}^\star_n$, and requires the sums of their probabilities to be not larger than 1. To get a RUM R out of a solution to the LP (1) one can assign, for each $\pi \in \mathbf{S}^\star_n$, probability p_π to π. Recall that $(1 \succ \cdots \succ n)$ is the only permutation of $[n]$ not in $\mathbf{S}^\star_n$ (this particular permutation does not contribute to the probability that j beats i, for any $j > i$). The probability that the RUM R assigns to $(1 \succ \cdots \succ n)$ is then equal to $1 - \sum_{\pi\in\mathbf{S}^\star_n} p_\pi$.

Let us first rewrite the above primal LP as:

$$\begin{cases} \min \binom{n}{2}^{-1} \cdot \sum_{1\le i<j\le n} \epsilon_{i,j} & \\ L_{i,j} : \epsilon_{i,j} + \sum_{\substack{\pi\in\mathbf{S}^\star_n \\ i\prec_\pi j}} p_\pi \ge P_{i,j} & \forall 1 \le i < j \le n \\ U_{i,j} : \epsilon_{i,j} - \sum_{\substack{\pi\in\mathbf{S}^\star_n \\ i\prec_\pi j}} p_\pi \ge -P_{i,j} & \forall 1 \le i < j \le n \\ D : -\sum_{\pi\in\mathbf{S}^\star_n} p_\pi \ge -1 & \\ \epsilon_{i,j} \ge 0 & \forall 1 \le i < j \le n \\ p_\pi \ge 0 & \forall \pi \in \mathbf{S}^\star_n \end{cases} \tag{2}$$

We then take its dual in order to reduce the number of variables to a polynomial in n:

$$\begin{cases} \max -D + \sum_{1\le i<j\le n} (P_{i,j} \cdot (L_{i,j} - U_{i,j})) & \\ \epsilon_{i,j} : L_{i,j} + U_{i,j} \le \binom{n}{2}^{-1} & \forall 1 \le i < j \le n \\ p_\pi : -D + \sum_{\substack{1\le i<j\le n \\ i\prec_\pi j}} (L_{i,j} - U_{i,j}) \le 0 & \forall \pi \in \mathbf{S}^\star_n \\ D, U_{i,j}, L_{i,j} \ge 0 & \forall 1 \le i < j \le n \end{cases} \tag{3}$$

Observe that each feasible solution to the Dual LP (3) can be transformed into a feasible solution with the same value and with the property that for each $i < j$ at most one of $U_{i,j}$ and $L_{i,j}$ is non-zero. (Indeed, if they are both positive, then we can subtract $\min(U_{i,j}, L_{i,j})$ from both of them without impacting feasibility and without changing the objective function's value.) Then, given a feasible solution to the Dual LP, we define $\Delta_{i,j} = U_{i,j} - L_{i,j}$ and after the above transformation, we are guaranteed that $U_{i,j} = \max(\Delta_{i,j}, 0)$, $L_{i,j} = \max(-\Delta_{i,j}, 0)$, and $|\Delta_{i,j}| = L_{i,j} + U_{i,j}$. The Dual LP (3) is then equivalent to:

$$\begin{cases} \max -D - \sum_{1\le i<j\le n} (P_{i,j}\cdot\Delta_{i,j}) & \\ -D - \sum_{\substack{1\le i<j\le n \\ i\prec_\pi j}} \Delta_{i,j} \le 0 & \forall \pi\in\mathbf{S}_n^\star \\ -\binom{n}{2}^{-1} \le \Delta_{i,j} \le \binom{n}{2}^{-1} & \forall 1\le i<j\le n \\ D\ge 0 & \end{cases} \tag{4}$$

Observe that LP (4) optimizes over a vector $\overline{\Delta}$ whose ℓ_∞-norm is bounded by $\binom{n}{2}^{-1}$. The goal is to maximize the objective function under the constraint that the transitive tournament whose (i,j) arc has weight $\Delta_{i,j}$, for each $1 \le i < j \le n$, has a minimum FAS of value at least $-D$. In other words, the separation oracle problem for LP (4) is an instance of Problem 2, i.e., of minimum FAS on $1/\binom{n}{2}$-bounded transitive tournaments.

We now transform LP (4) from a maximization problem into a feasibility one, since we will be using the Ellipsoid algorithm to solve it:

$$F_\rho := \begin{cases} c_\rho : -D - \sum_{1\le i<j\le n} (P_{i,j}\cdot\Delta_{i,j}) \ge \rho & \\ c_\pi : -D - \sum_{\substack{1\le i<j\le n \\ i\prec_\pi j}} \Delta_{i,j} \le 0 & \forall\pi\in\mathbf{S}_n^\star \\ c_{i,j} : -\binom{n}{2}^{-1} \le \Delta_{i,j} \le \binom{n}{2}^{-1} & \forall 1\le i<j\le n \\ c_D : D\ge 0 & \end{cases}$$

Clearly, LP (4) has value at least ρ iff F_ρ is feasible. We now give an approximate separation oracle for F_ρ.

Theorem 8. *Fix any $\delta(n) \ge \Omega\left(\log^{-0.49} n\right)$. Then, there exists a randomized algorithm that gets as input an assignment $\{D, \Delta_{1,2}, \ldots, \Delta_{n-1,n}\}$ to F_ρ, and that in time $O(n^2)$, with probability at least $2/3$, (i) returns an unsatisfied constraint of F_ρ if at least one of the constraints c_ρ, c_D, or $c_{i,j}$ (for $1 \le i < j \le n$) is unsatisfied, otherwise (ii) if there exists at least one $\pi \in \mathbf{S}_n^\star$ such that $-D - \sum_{\substack{1\le i<j\le n \\ i\prec_\pi j}} \Delta_{i,j} > \delta(n)$, the algorithm returns an unsatisfied c_π constraint, otherwise (iii) the algorithm might not return any unsatisfied constraint (even if some exists).*

Moreover, if there exists at least one $\pi \in \mathbf{S}_n^\star$ such that $-D - \sum_{\substack{1\le i<j\le n \\ i\prec_\pi j}} \Delta_{i,j} > \delta(n)$, then for any constant $c > 0$, if the above randomized algorithm is run independently for $\Theta(c\log n)$ times, the probability that no unsatisfied constraint is returned shrinks to n^{-c}.

Proof. The algorithm can check the validity of each of the c_ρ, c_D, and $c_{i,j}$ (for $1 \le i < j \le n$) constraints in time $O(n^2)$; if one of the constraints is unsatisfied, it can be returned. Otherwise, they are all valid, and the algorithm creates the transitive tournament $G(V, A, w)$ with $V = [n]$, $A = \{(i,j) \mid 1 \le i < j \le n\}$, and $w((i,j)) = \Delta_{i,j}$. Then, $G(V, A, w)$ is a τ-bounded transitive tournament, for $\tau \le O(n^{-2})$, i.e., is an instance of Problem 2. In fact, consider any $\pi \in \mathbf{S}_n^\star$: if $C''_{G(V,A,w)}(\pi)$ is the cost of solution π for the instance $G(V, A, w)$ of Problem 2, then the constraint c_π of F_γ is exactly $-D - C''_{G(V,A,w)}(\pi) \le 0$, or equivalently, $C''_{G(V,A,w)}(\pi) \ge -D$.

Theorem 3 and Observation 4 imply the existence of an algorithm for Problem 2 on $G(V, A, w)$ that additively approximates the optimal solution to any $\delta > 0$ in time $O\left(n\cdot\delta^{-4}\right) + 2^{\widetilde{O}(\delta^{-2})}$. In particular, for each $\delta(n) \ge \Omega\left(\log^{-0.49} n\right)$, the problem can be approximated to within an additive $\delta(n)$ in time $O(n^2)$.

Suppose that π' is the permutation returned by the additive-approximation algorithm. Then, with probability at least $2/3$, for each $\pi \in \mathbf{S}_n$ it holds that $C''_{G(V,A,w)}(\pi) \ge C''_{G(V,A,w)}(\pi') - \delta(n)$.

Now, if $-D - C''_{G(V,A,w)}(\pi') > 0$ then $c_{\pi'}$ is an unsatisfied constraint, which can be returned (indeed, if this happens, then $\pi' \in \mathbf{S}_n^\star$; in fact, $1 \succ 2 \succ \cdots \succ n$ is the only permutation in $\mathbf{S}_n - \mathbf{S}_n^\star$, and $C''_{G(V,A,w)}$ has a value of 0 on this permutation). Otherwise, $C''_{G(V,A,w)}(\pi') \ge -D$; then, it must hold that $C''_{G(V,A,w)}(\pi) \ge C''_{G(V,A,w)}(\pi') - \delta(n) \ge -D - \delta(n)$, i.e., for each $\pi \in \mathbf{S}_n^\star$, the constraint $-D - \sum_{\substack{1\le i<j\le n \\ i\prec_\pi j}} \Delta_{i,j} \le \delta(n)$ must hold.

Finally, to decrease the error probability to n^{-c}, it suffices to (independently) run the randomized algorithm for Problem 2, $O(c\log n)$ times. Then a permutation with the smallest cost $C''_{G(V,A,w)}$ in these runs can be returned. □

We can now use the approximate separation oracle of Theorem 8 along with the Ellipsoid algorithm (Grötschel et al., 1988), to obtain a δ-additive approximation to Problem 6.

Theorem 9. *Problem 6 can be additively approximated to any $\delta \ge \Omega(\log^{-0.49} n)$ in polynomial time.*

Proof. First, our algorithm RIPPLE guesses a $\rho \in \left\{i\cdot\delta \mid 0 \le i \le \left\lfloor \frac{1}{\delta} \right\rfloor\right\}$ (the algorithm can binary search among the values in this set). For a given ρ, the Ellipsoid algorithm (Grötschel et al., 1988) can be used with

the approximate separation oracle of Theorem 8 to approximately check the non-emptiness of F_ρ. In particular, the Ellipsoid algorithm will call the separation oracle at most polynomially many times, obtaining at most polynomially many separating hyperplanes. If the set of returned hyperplanes forms an infeasible LP, the Ellipsoid algorithm can correctly conclude that F_ρ is empty.

Otherwise, the Ellipsoid algorithm could return a point $x = (D, \Delta_{1,2}, \ldots, \Delta_{n-1,n})$ that the oracle was unable to separate from F_ρ. This point could be outside of F_ρ since the oracle only guarantees[6] that the c_π constraints hold to within an additive δ error. We then define the point $x' = (D + \delta, \Delta_{1,2}, \ldots, \Delta_{n-1,n})$; we have that $x' \in F_{\rho-\delta}$. Indeed, (i) each c_π constraint will be satisfied by x' (with x, c_π is off by at most δ; and, the value of the LHS of the c_π constraint decreases by δ from x to x'), (ii) the $c_{\rho-\delta}$ constraint of $F_{\rho-\delta}$ will be satisfied by x' (the constraint c_ρ of F_ρ is satisfied with x, thus $c_{\rho-\delta}$ is satisfied by x'), and (iii) each remaining constraint will still be satisfied.

Suppose that $i^\star$ is the smallest i for which the algorithm could conclude that $x' \in F_{\rho^\star-\delta}$ where $\rho^\star = i^\star \cdot \delta$.[7]

Then, the Dual LP (4) does not admit a solution of value at least $\rho^\star + \delta$, but admits a solution of value at least $\rho^\star - \delta$. It follows that the optimal solution of the Dual LP (4), and thus of the Primal LP (1), lies in $[\rho^\star - \delta, \rho^\star + \delta]$. Hence, the Ellipsoid algorithm with the above separation oracle, lets us additively approximate the value of the optimal solution of Problem 6 to within 2δ.

To complete the description of RIPPLE, it only remains to obtain an approximating RUM having an error within the smallest possible plus 2δ. 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; thus, the oracle returns at most polynomially many separating hyperplanes, some of which might refer to non-permutation constraints, while the rest refer to the permutation constraints of, say, permutations $\pi_1, \ldots, \pi_t$ ($t \leq n^{O(1)}$). Now, if we restrict the primal LP (1) to (i) its non-permutation variables and (ii) the permutation variables $p_{\pi_1}, \ldots, p_{\pi_t}$, then we obtain an LP of size polynomial in n (hence, solvable in polynomial time) and with an optimal value not larger than 2δ plus the optimum of the primal LP (1). Thus, solving the restricted LP allows us to obtain a RUM with an error not larger than the smallest possible plus 2δ. □

One might ask if the same technique can be used to minimize uniform error as in Definition 7. Unfortunately, the separation oracle for this version is NP-hard to approximate to some additive constant. We discuss the uniform error problem in Section B of the Supplementary Material.

5. Succinct Representation for Average Errors

The RUMs obtained from Theorem 9 might be supported by (polynomially) many permutations. In this section we discuss succinct representation of those (and other) RUMs.

First, we note that a result of Chierichetti et al. (2021) implies that any RUM can be sketched to $O(\epsilon^{-2} \cdot n \log^2 n)$ bits in such a way that the probability distribution of *each* slate of size two is approximated to within an additive error ϵ. Here, we will show that $O(\epsilon^{-2} \cdot n \log n)$ bits suffice to approximate the average ℓ_1-error (and the average ℓ_2-error) over slates of size two to within an additive error ϵ.

As we will see, our sketch is a uniform RUM supported on a multiset of $O(\epsilon^{-2})$ permutations, i.e., by a number of permutations that is independent of n. Note that a uniform RUM supported on k permutations can be represented by a k-dimensional vector per item; the generic item's vector contains the ranks of that item in the k permutations of the RUM. Therefore, our succinct representation can also be seen as an embedding of the items in an $O(\epsilon^{-2})$-dimensional space.

Define the (pairwise) *RMSE*[8] between RUMs D and D' as

$$\rho(D, D') = \sqrt{\frac{\sum_{i,j \in [n], i \neq j} \left(D_{\{i,j\}}(j) - D'_{\{i,j\}}(j) \right)^2}{n(n-1)}}.$$

We also define a ℓ_q-version of the above measure:

$$\rho_q(D, D') = \left(\binom{n}{2}^{-1} \sum_{i,j \in [n], i<j} \left| D_{\{i,j\}}(j) - D'_{\{i,j\}}(j) \right|^q \right)^{\frac{1}{q}}$$

Since $D_{\{i,j\}}(i) = 1 - D_{\{i,j\}}(j)$, we get $|D_{\{i,j\}}(j) - D'_{\{i,j\}}(j)| = |D_{\{i,j\}}(i) - D'_{\{i,j\}}(i)|$; thus, $\rho = \rho_2$.

The sketch we propose samples $O(\epsilon^{-2})$ permutations from the original RUM D and uses them as the support of a new RUM $\widetilde{D}$ that, as we show, approximates D both in the ρ_1- and ρ_2-senses.

Theorem 10. *For a RUM D and for each $\epsilon > 0$, there is a RUM $\widetilde{D}$ that samples uniformly from a multiset of $O\left(\epsilon^{-2}\right)$*

[6] Given that the approximate separation oracle is a randomized algorithm, this guarantee might fail to hold. Still, given that the oracle is called no more than polynomially many times, and given that the probability of its error can be made as small as n^{-c}, for an arbitrary constant $c > 0$, with high probability the guarantee will hold at each oracle call.

[7] Such an $i^\star$ must exist since Problem 6 always admits a solution of value at most $1/2$. Indeed, the RUM such that $R_{\{i,j\}}(j) = 1/2$, for each $1 \leq i < j \leq n$, can be realized, e.g., with a uniform distribution over $\mathbf{S}_n$.

[8] Note that the RMSE defined by Makhijani & Ugander (2019) is slightly different: their denominator is n^2 instead of $n \cdot (n-1)$. It is possible to show that our RUM approximation holds for their RMSE version as well.

Algorithm 1 RUMRUNNER, a heuristic for Problem 6.

$S \leftarrow \varnothing$
$\pi^\star \leftarrow (1 \prec \cdots \prec n)$
repeat
  $S \leftarrow S \cup \{\pi^\star\}$
  Solve the primal LP (1) restricted to the variables ϵ, and p_π for $\pi \in S$; let $\mathcal{P}$ be its optimal primal solution, and $\mathcal{D}$ be its optimal dual solution
  $\pi^\star \leftarrow$ `Viol-HP` $(\mathcal{D})$
until $\pi^\star = \perp$
return the RUM induced by $\mathcal{P}$, i.e., the RUM that samples $\pi \in S$ with probability $\mathcal{P}(p_\pi)$

permutations such that $\rho_1(D, \widetilde{D}) < \epsilon$ and $\rho_2(D, \widetilde{D}) < \epsilon$.

Since each permutation can be represented with $O(n \log n)$ bits, an immediate consequence of Theorem 10 is that all RUMs (in particular, those obtained by our algorithms) can be represented succinctly.

Corollary 11. *For a RUM D on $[n]$ and for each $\epsilon > 0$, there is a RUM $\widetilde{D}$ representable with $O(\epsilon^{-2} \cdot n \log n)$ bits such that $\rho_1(D, \widetilde{D}) < \epsilon$ and $\rho_2(D, \widetilde{D}) < \epsilon$.*

6. Experiments

We present three groups of experiments. Section 6.3 studies the effectiveness of RUMRUNNER at representing a given tournament matrix. As described in the Introduction, we view this as the key metric for situations in which the tournament matrix is known to reasonable precision, and we find that the approximation error is very small for all datasets (zero, or order of $1/n^2$ for an $n \times n$ tournament).

Next, in Section 6.4, we ask whether the resulting RUM can be approximated by a much smaller RUM; we consider a number of heuristics to sketch a RUM with support on only $k = 10$ permutations, and identify two approaches that are able to approximate all tournament matrices well in this regime. This suggests that very small RUMs may approximate real-world datasets, resulting in computational efficiencies and increased transparency.

Finally, in Section 6.5, we consider a standard prediction setting in which the data is split between train/test, and the learned model is measured according to its generalization performance. This setting is appropriate if the training sample is too small to fully characterize the data distribution.

6.1. Linear programs and separation oracles

For computational efficiency reasons, we do not use RIPPLE as described to fit RUMs to the datasets. Instead, we develop RUMRUNNER using a separating-hyperplane approach, described in Algorithm 1.

RUMRUNNER employs a separation oracle for the dual: a function `Viol-HP` that, given the transitive tournament induced by the dual solution, either returns a violated hyperplane, or $\perp$ if it could not find any such hyperplane. As mentioned in Section 4, the `Viol-HP` separation oracle needs to find a permutation π inducing a minimum FAS of the transitive tournament. Given such a π, the function needs only to check if $\sum_{i<j, i \prec_\pi j} \mathcal{D}(\Delta_{i,j}) \geq -\mathcal{D}(D)$.

Algorithm 2 A randomized local search for `Viol-HP`, i.e., the separation oracle. In experiments, we set $t = 100$.

For a permutation π, let (i) $\mathrm{fas}(\pi) = \sum_{i<j \atop i \prec_\pi j} \mathcal{D}(\Delta_{i,j})$ be the FAS cost of π on the dual solution's directed graph and (ii) $N(\pi)$ be the set of permutations that can be obtained by moving *one* of the elements of π to a new position.
$\mathrm{fas}_{\min} \leftarrow \infty$
for $i = 1, \ldots, t$ **do**
  $\pi \leftarrow$ uniform at random permutation from $\mathbf{S}_n$
  while $\exists \pi' \in N(\pi)$ such that $\mathrm{fas}(\pi') < \mathrm{fas}(\pi)$ **do**
    $\pi \leftarrow \arg\min_{\pi' \in N(\pi)} \mathrm{fas}(\pi')$
  if $\mathrm{fas}(\pi) < \mathrm{fas}_{\min}$ **then**
    $\mathrm{fas}_{\min} \leftarrow \mathrm{fas}(\pi)$
    $\pi_{\min} \leftarrow \pi$
if $-\mathcal{D}(D) - \mathrm{fas}_{\min} \leq 0$ **then**
  return $\perp$
else
  return $\pi_{\min}$

Now, given that finding a minimum FAS is NP-hard, we implemented two versions of `Viol-HP`. One is the classical exact dynamic programming algorithm (Lawler, 1964), running in time $O(n2^n)$, and another is a randomized local search heuristic, Algorithm 2; the latter might fail to return the minimum FAS. In our experiments, we could run the exact algorithm on most of the datasets; for the remaining, we resorted to Algorithm 2, sampling 100 permutations, and stopping an iteration if the gain in the last step was less than 10^{-5}. Note that Algorithm 2 might fail to find a separating hyperplane even if it exists, which means RUMRUNNER might fail to return a RUM at the minimum distance from the input matrix. (This is the reason why Table 1 has two columns detailing the average-error of RUMs: the first column gives the average-error that the RUM returned by RUMRUNNER has with respect to the matrix, the second gives a lower bound on the minimum possible average-error achievable by a RUM with respect to the matrix.) In practice however, as we discuss in Section C of the Supplementary Material, RUMRUNNER works fairly well: it always returns a RUM (hence always gives a correct upper bound on the distance of the input matrix to RUMs) and, with an extra final check on the feasibility for the dual of its last $\mathcal{D}$ solution, RUMRUNNER can also certify a lower bound on the distance of the input matrix to the best approximating RUM. In our experiments, this final check was implemented with an exact algorithm.

Table 1. Statistics about the RUMs obtained by RUMRUNNER: the cardinality of the support ($|S|$), the average error, a lower bound on the average error, and the uniform error.

Dataset	n	$\lvert S\rvert$	avg. err.	lower bound on avg. err.	unif. err.
A5	16	121			
A9	12	67			
A17	13	79		0	
A48	10	46			
A81	11	56			
SF	35	572	0.001408	0.001408	0.1438
Jester	100	3553	0.000461	0	0.0786

6.2. Experimental Setup

Our main algorithm is RUMRUNNER, which uses Algorithm 1 along with either Algorithm 2 or the exact algorithm (Lawler, 1964) for `Viol-HP`. We implemented RUMRUNNER in Python using IBM cplex[9] as the LP solver. All experiments were done on commodity hardware[10].

Baselines. As baselines we considered the Thurstone model (Thurstone, 1994), Multinomial Logit (MNL) (also known as Bradley–Terry–Luce model) (Bradley & Terry, 1952), Blade-Chest (BC) (Chen & Joachims, 2016), and 3D Gaussian majority vote (MV) (Makhijani & Ugander, 2019). We used, as implementations of these baselines, those made available by Chen & Joachims (2016) and Makhijani & Ugander (2019).

Datasets. We measured the performances of the algorithms on a number of datasets: a class of commonly studied *election* datasets `A5`, `A9`, `A17`, `A48`, `A81` (Tideman, 2006), one videogame dataset representing matchups between *Super Street Fighter IV* (`SF`) characters (Chen & Joachims, 2016), and the `Jester` rating dataset (Goldberg et al., 2001).

Unlike the others, the election datasets are composed of ballots, each containing a partial ranking (i.e., a ranking of a subset) of the candidates. Following Makhijani & Ugander (2019), as we are looking for pairwise matches, we "transformed" a generic ballot—a ranking of a subset S of candidates—into a set of $\binom{|S|}{2}$ matches, one for each pair of candidates in S, where the winner of the generic match is the candidate that is ranked higher in the ballot.

6.3. RUM Approximation

Table 1 shows how close the RUMs computed by RUMRUNNER come to their respective input tournament matrices. In the case of the election datasets the error is zero: the representation is perfect. For `SF`, the total error $= \binom{35}{2} \cdot 0.001408\ldots \approx 0.837$, which is smaller than 1, the range of an entry of the 35×35 matrix. For `Jester`, the best RUM we could find induces a total error of $\binom{100}{2} \cdot 0.000461\ldots \approx 2.281$, i.e., slightly more than two entries of the 100×100 matrix. While the algorithms are crafted to minimize average error, we also report the uniform/maximum error.

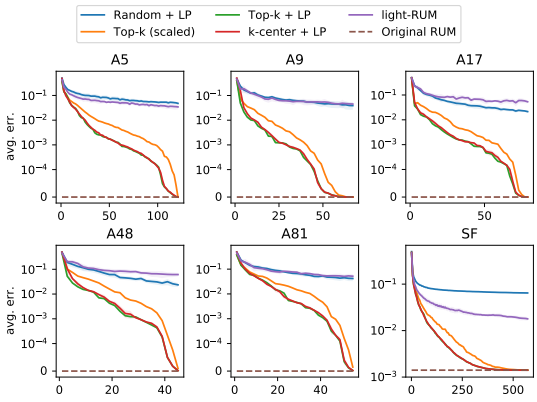

Figure 1. Average error for different number of permutations k (horizontal axis) for the considered heuristics.

6.4. Sketching

Theorem 10 shows that the support of a generic RUM can be reduced to $O(\epsilon^{-2})$ permutations while guaranteeing that the new RUM is within a distance ϵ to the original, both in the RMSE and in the average sense. In this section we describe two heuristics for support reduction, that appear to work significantly better in practice. We apply our heuristics to the RUMs produced by our algorithm; we note, though, that they can be applied to any RUM.

For each of our RUMs R, and for a fixed value of k, we considered the following heuristics:

- *Random + LP*: sample a set of k permutations u.a.r. from $\mathbf{S}_n$, and run the primal LP (1) on that set of permutations to obtain their weights.
- *Top-k (scaled)*: consider the set T of the k permutations having largest probability in R; create a new RUM supported on T, assign probabilities to T's permutations by scaling up their probabilities in R.
- *Top-k + LP*: given T as above, run the primal LP (1) on T to assign new probabilities to its permutations.
- *k-Center + LP*: run the Greedy k-center algorithm of (Gonzalez, 1985) on the support permutations of the RUM, using their Kendall distance scaled by the product of their probabilities in R; run the LP on the returned centers to assign them probabilities.
- *Light-RUM*: sample k permutations from R as in Theorem 10.

Each heuristic returns a RUM supported on $\leq k$ permutations. We compared these RUMs to the tournament matrix that was used to produce R, i.e., to the original dataset.

[9] `http://ibm.com/analytics/cplex-optimizer`
[10] 12-Core Ryzen 3900X with 64GB of RAM

Table 2. Average error of RUMs obtained through different support reduction heuristics, with the target support of size $k = 10$.

Dataset	Original RUM	*Random + LP*	*Top-k (scaled)*	*Top-k + LP*	*k-center + LP*	*Light-RUM*
A5	0	0.156 ±0.020	0.059	**0.041**	0.045 ±0.004	0.111 ±0.016
A9	0	0.133 ±0.030	0.031	**0.010**	0.011 ±0.002	0.119 ±0.023
A17	0	0.112 ±0.014	0.034	**0.014**	**0.014 ±0.001**	0.119 ±0.033
A48	0	0.107 ±0.022	0.043	**0.015**	**0.015 ±0.001**	0.119 ±0.029
A81	0	0.129 ±0.024	0.056	0.047	**0.046 ±0.005**	0.113 ±0.021
SF	0.00141	0.149 ±0.008	0.113	**0.104**	0.105 ±0.008	0.127 ±0.011
Jester	0.00046	0.168 ±0.008	0.119	**0.108**	**0.108 ±0.003**	0.121 ±0.006

In Table 2, we report the (expected) average error of the different heuristics (each averaged over 20 runs) for a fixed value of $k = 10$. In Figure 1, we report the (expected) average error for different values of k. Observe that the quality of the solution produced by the LP (1) has a significant dependence on the set of permutations/variables available to it—*Random + LP* is significantly worse than the other heuristics that use the LP in their final step. Still, if we limit ourselves to scaling up the probabilities of the top-k permutations of the original RUM (*Top-k (scaled)*), the results are worse than, but comparable to, those obtained by the best LP method. It is also clear that *Light-RUM* is penalized by its uniformity—its probabilities, by design, have a $1/k$ granularity.

Even with $k = 10$ permutations, the RUMs obtained by the two best heuristics, *Top-k + LP* and *k-center + LP* closely approximate the original tournament matrix.

6.5. Quality of generalization

We also compared the predictive performance of our algorithm with respect to several baselines. Our tests follows those of Makhijani & Ugander (2019): we use a five-fold cross-validation, and measure their RMSE as a metric.

For each dataset, we first compute the list of its t matchups. We then split, uniformly at random, the list into five parts, conditioned on each part having length equal to either $\lfloor t/5 \rfloor$ or $\lceil t/5 \rceil$. Then, for each part, we considered that part as the test set, and the union of the remaining four parts as the training set. For each train/test pair, we ran all the algorithms on the training data to obtain a prediction matrix. Finally, we computed the RMSE as defined in Makhijani & Ugander (2019), $\rho'(D, D') = \sqrt{\sum_{i,j \in [n], i \neq j} \left(D_{\{i,j\}}(j) - D'_{\{i,j\}}(j)\right)^2 / n^2}$, between each prediction matrix and its corresponding test set. We randomly sampled 10 splittings obtaining, for each dataset, 50 train/test pairs.

In Table 3 we report our prediction results. Observe that the baseline results are different from the ones reported in Makhijani & Ugander (2019),[11] but the ratios of their RMSEs remain similar.

The results show that RUMRUNNER is competitive with the other baselines. Since we are able to approximate perfectly (e.g., in the elections datasets) or almost perfectly (SF) the training matrix, our algorithm is essentially obtaining the same results of the training matrix used, with no change, as the predictor.

Table 3. Expected RMSE in the cross-validation experiment. We sampled ten 5-fold splittings per dataset. The only significant difference is that RUMRUNNER beats MNL and Thurstone for A9. The standard deviations are all in the range $[\pm 0.002, \pm 0.007]$.

Dataset	MNL	Thurstone	BC	MV	RUMRUNNER
A5	0.041	0.041	0.041	0.041	0.043
A9	0.026	0.027	0.023	0.023	0.021
A17	0.049	0.050	0.051	0.050	0.052
A48	0.034	0.035	0.033	0.034	0.035
A81	0.043	0.043	0.042	0.042	0.043
SF	0.137	0.137	0.138	0.137	0.142

7. Conclusions

In this paper we studied the problem of approximating a given tournament matrix by a RUM. We obtained an algorithm to minimize the average error to within a $o(1)$ margin; we also showed that such an approach is unlikely to work to minimize the uniform error. A natural open question is to extend our methods beyond head-to-head contests. For example, given the winning probabilities in slates of size three, what is the complexity of finding the closest RUM, to minimize the average error? Another interesting question is the following: given a tournament matrix, is there a RUM on support of size k that exactly represents it?

Acknowledgments

We thank the anonymous reviewers for their careful reading. We also thank Mohammad Mahdian, Rahul Makhijani, Tim Roughgarden, Johan Ugander, and Santosh Vempala 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 PRIN project 2017K7XPAN.

[11] We confirmed with the authors that our numbers are indeed correct; the discrepancy is due to a library error.

References

Adams, R. P., Dahl, G. E., and Murray, I. Incorporating side information in probabilistic matrix factorization with Gaussian processes. In *UAI*, pp. 1–9, 2010.

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., and Cohen, J. The physics of optimal decision making: A formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological Review*, 113(4):700–765, 2006.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Chen, S. and Joachims, T. Modeling intransitivity in matchup and comparison data. In *WSDM*, pp. 227–236, 2016.

Chierichetti, F., Kumar, R., and Tomkins, A. Discrete choice, permutations, and reconstruction. In *SODA*, pp. 576–586, 2018a.

Chierichetti, F., Kumar, R., and Tomkins, A. Learning a mixture of two multinomial logits. In *ICML*, pp. 961–969, 2018b.

Chierichetti, F., Kumar, R., and Tomkins, A. Light RUMs. In *ICML*, pp. 1888–1897, 2021.

Clementi, A. and Trevisan, L. Improved non-approximability results for minimum vertex cover with density constraints. *TCS*, 225(1):113–0128, 1999.

Farias, V. F., Jagabathula, S., and Shah, D. A data-driven approach to modeling choice. In *NIPS*, pp. 504–512, 2009.

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

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4:133–151, 07 2001.

Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985.

Grötschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer, 1988.

Karp, R. M. Reducibility among combinatorial problems. In Miller, R. E., Thatcher, J. W., and Bohlinger, J. D. (eds.), *Complexity of Computer Computations. The IBM Research Symposia Series*, pp. 85–103. Springer, 1972.

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

Makhijani, R. and Ugander, J. Parametric models for intransitivity in pairwise rankings. In *WWW*, pp. 3056–3062, 2019.

Negahban, S., Oh, S., Thekumparampil, K. K., and Xu, J. Learning from comparisons and choices. *JMLR*, 19(1): 1478–1572, 2018.

Oh, S. and Shah, D. Learning mixed multinomial logit model from ordinal data. In *NIPS*, pp. 595–603, 2014.

Soufiani, H. A., Parkes, D. C., and Xia, L. Random utility theory for social choice. In *NIPS*, pp. 126–134, 2012.

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

Thurstone, L. L. A law of comparative judgment. *Psychological Review*, 34:273–286, 1994.

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

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

Veerathu, V. and Rajkumar, A. On the structure of parametric tournaments with application to ranking from pairwise comparisons. In *NeurIPS*, 2021.

Supplementary Material

A. Missing Proofs

A.1. Proof of Observation 4

Proof. Let $G = ([n], A', w')$ be an instance of Problem 1. Let $A'' = \{(i,j) \mid 1 \le i < j \le n\}$, and for each $1 \le i < j \le n$, let

$$w''((i,j)) = \begin{cases} w'((i,j)) & \text{if } (i,j) \in A', \\ -w'((i,j)) & \text{if } (j,i) \in A', \\ 0 & \text{otherwise.} \end{cases}$$

Then, $G = (V, A'', w'')$ is an instance of Problem 2. Now, let $W' = \sum_{\substack{(i,j)\in A' \\ j>i}} w'((i,j))$; then, for each permutation $\pi \in \mathbf{S}_n$, it holds that $C''_{G=(V,A'',w'')}(\pi) = C'_{G=(V,A',w')}(\pi) - W'$. Thus additive approximation is preserved in this direction.

Analogously, let $G = ([n], A'', w'')$ be an instance of Problem 2. Let $A' = \{(i,j) \mid (1 \le i < j \le n \text{ and } w''((i,j)) \ge 0)$ or $(1 \le j < i \le n \text{ and } w''((i,j)) < 0)\}$, and for each $(i,j) \in A'$, let

$$w'((i,j)) = \begin{cases} w''((i,j)) & \text{if } i < j, \\ -w''((i,j)) & \text{if } j < i. \end{cases}$$

Then, $G = (V, A', w')$ is an instance of Problem 1. Now, let $W'' = \sum_{\substack{(i,j)\in A'' \\ w''((i,j))<0}} w''((i,j))$; then, for each permutation $\pi \in \mathbf{S}_n$, it holds that $C'_{G=(V,A',w')}(\pi) = C''_{G=(V,A'',w'')}(\pi) - W''$. Thus additive approximation is preserved in this other direction, as well. □

A.2. Proof of Theorem 10

Proof. In a manner similar to Chierichetti et al. (2021), we sample $k = \left\lceil \frac{4}{\epsilon^2} \right\rceil$ independent permutations $\pi_1, \ldots, \pi_k$ from D, and we let $\widetilde{D}$ be the uniform RUM over the multiset $\{\pi_1, \ldots, \pi_k\}$, i.e., the RUM $\widetilde{D}$ samples i uniformly at random from $[k]$, and returns π_i.

For arbitrary $i < j$, consider the random variable $X = \widetilde{D}_{\{i,j\}}(j)$. Then, $X = \frac{k'}{k}$ iff there are exactly k' indices t in $[k]$ such that $j \succ_{\pi_t} i$. Observe that, by the iid choice of the π_i's from D, the distribution of k' is equal to the binomial distribution $\mathrm{Bin}(k,p)$ with $p = D_{\{i,j\}}(j)$.

Then, $\mathrm{E}[X] = \mathrm{E}\left[\widetilde{D}_{\{i,j\}}(j)\right] = \frac{kp}{k} = p = D_{\{i,j\}}(j)$, and

$$\mathrm{E}\left[(X-p)^2\right] = \mathrm{Var}[X] = \frac{kp(1-p)}{k^2} = \frac{p(1-p)}{k} \le \frac{1}{4k},$$

since $p(1-p)$ is maximized at $p = 1/2$.

Using Jensen's inequality, $\mathrm{E}[|X-p|] \le \sqrt{\mathrm{E}[(X-p)^2]}$,

$$\mathrm{E}\left[\left|\widetilde{D}_{\{i,j\}}(j) - D_{\{i,j\}}(j)\right|\right] = \mathrm{E}[|X-p|] \le \frac{1}{2\sqrt{k}}.$$

Thus for each $q \in \{1,2\}$,

$$\mathrm{E}\left[\left|\widetilde{D}_{\{i,j\}}(j) - D_{\{i,j\}}(j)\right|^q\right] \le (4k)^{-q/2}.$$

Let

$$S_q = \sum_{1\le i<j\le n} \left|\widetilde{D}_{\{i,j\}}(j) - D_{\{i,j\}}(j)\right|^q.$$

By the linearity of expectation, we have

$$\mathrm{E}[S_q] \le \binom{n}{2}(4k)^{-q/2}.$$

We also have $S_q \ge 0$; thus, by Markov's inequality, we get that $\Pr[S_q < 4\mathrm{E}[S_q]] > 3/4$. Consider the event $\xi_q =$"$S_q <$

$\binom{n}{2}4^{1-q/2} \cdot k^{-q/2}$". We get

$$\Pr[\xi_q] \geq \Pr[S_q < 4\mathrm{E}[S_q]] > \frac{3}{4}.$$

Now, $\rho_q\left(D, \widetilde{D}\right) = \sqrt[q]{S_q/\binom{n}{2}}$. Then, ξ_q entails that

$$\rho_q\left(D, \widetilde{D}\right) < \sqrt[q]{\frac{\binom{n}{2}4^{1-\frac{q}{2}} \cdot k^{-\frac{q}{2}}}{\binom{n}{2}}} \leq 4^{\frac{1}{q}-\frac{1}{2}} \cdot k^{-\frac{1}{2}} \leq \epsilon,$$

where the last inequality follows from $q \geq 1$ and $k \geq \frac{4}{\epsilon^2}$. Since $\Pr[\xi_q] > 3/4$ for $q \in \{1, 2\}$, with probability $1/2$, $\widetilde{D}$ is an ϵ-approximation of D both in ρ_1- and ρ_2-senses. □

B. Minimizing Uniform Error

As in the average error case, once can write the primal LP for minimizing the uniform error:

$$\begin{cases} \min \epsilon \\ L_{i,j} : \epsilon + \sum\limits_{\substack{\pi \in \mathbf{S}_n^\star \\ i \prec_\pi j}} p_\pi \geq P_{i,j} & \forall 1 \leq i < j \leq n \\ U_{i,j} : \epsilon - \sum\limits_{\substack{\pi \in \mathbf{S}_n^\star \\ i \prec_\pi j}} p_\pi \geq -P_{i,j} & \forall 1 \leq i < j \leq n \\ D : -\sum\limits_{\pi \in \mathbf{S}_n^\star} p_\pi \geq -1 \\ \epsilon \geq 0 \\ p_\pi \geq 0 & \forall \pi \in \mathbf{S}_n^\star \end{cases}$$

We then take the dual, obtaining:

$$\begin{cases} \max -D + \sum\limits_{1 \leq i < j \leq n} (P_{i,j} \cdot (L_{i,j} - U_{i,j})) \\ \epsilon : \sum\limits_{1 \leq i < j \leq n} (L_{i,j} + U_{i,j}) \leq 1 \\ p_\pi : -D + \sum\limits_{\substack{1 \leq i < j \leq n \\ i \prec_\pi j}} (L_{i,j} - U_{i,j}) \leq 0 & \forall \pi \in \mathbf{S}_n^\star \\ D, U_{i,j}, L_{i,j} \geq 0 & \forall 1 \leq i < j \leq n \end{cases}$$

As before, we observe that if $\min(U_{i,j}, L_{i,j}) > 0$, then we can subtract this quantity from both $L_{i,j}$ and $U_{i,j}$ with no impact on feasibility nor on the objective. This way, we can set $\Delta_{i,j} = U_{i,j} - L_{i,j}$. The ϵ constraint then becomes $\sum_{1 \leq i < j \leq n} |\Delta_{i,j}| \leq 1$. Thus, the dual LP is equivalent to:

$$\begin{cases} \max -D - \sum\limits_{1 \leq i < j \leq n} (P_{i,j} \cdot \Delta_{i,j}) \\ -D - \sum\limits_{\substack{1 \leq i < j \leq n \\ i \prec_\pi j}} \Delta_{i,j} \leq 0 & \forall \pi \in \mathbf{S}_n^\star \\ |\overline{\Delta}|_1 \leq 1 \\ D \geq 0 \end{cases} \tag{5}$$

Thus, when minimizing the maximum error, the dual LP optimizes over a vector $\overline{\Delta}$ whose ℓ_1-norm is at most 1 — recall that, in the case of average error minimization, it was the ℓ_∞-norm of $\overline{\Delta}$ that was bounded to be at most 1. The goal of the LP does not change: this LP aims to maximize the same objective function of the LP for the average error, under the same constraint that the $\overline{\Delta}$-graph has a minimum FAS of value at least $-D$.

We show that the Separation Oracle problem for LP (5) is NP-hard to additively approximate to some positive constant. Hence, the Ellipsoid-based approach we used to get an $o(1)$-approximation algorithm for Problem 6, fails for the maximum-error variant.

Theorem 12. *There exists a constant $\alpha > 0$ such that, given an assignment to (5), it is NP-hard to determine whether the assignment satisfies each constraint, or whether some constraint is off by at least α.*

Proof. The classical reduction (Karp, 1972) from Vertex-Cover to (unweighted) Feedback Arc Set[12] creates a FAS instance composed of the digraph H with $2n$ vertices, max-indegree and max-outdegree not larger than $\delta + 1$, and a number of arcs equal to $2n + 2m$, starting from a Vertex Cover instance composed of a graph G, having n nodes, maximum degree δ, and m edges. The reduction guarantees the existence of a k-Vertex Cover in $G(V, E)$ iff a k-Feedback Arc Set exists in $H(V, A)$.

The Vertex Cover instances of Clementi & Trevisan (1999) have maximum degree $\delta \leq c = O(1)$, so that $m \leq \frac{\delta}{2} \cdot n$ and, also, $m \geq \frac{n}{2}$;[13] Clementi & Trevisan (1999) show that there exists a constant $c' > 1$ such that it is NP-hard to distinguish whether the minimum Vertex Cover of such an instance has size $\leq k$ or $\geq c'k$, for $k \geq \frac{m}{\delta}$.[14]

Then, if we plug the Vertex Cover instances of Clementi & Trevisan (1999) into the reduction of Karp (1972), we get a digraph H having $m' = 2n + 2m$ arcs, with

$$m' \leq 2n + 2 \cdot \frac{\delta}{2} \cdot n = 2n + \delta n \leq (2 + c) \cdot n,$$

and such that it is NP-hard to tell whether the minimum FAS of H is smaller than k or larger than $c'k$, for the inapproximability ratio $c' > 1$ of (Clementi & Trevisan, 1999).

We observe that, by $k \geq \frac{m}{\delta}$, $m \geq \frac{n}{2}$ and $\delta \leq c$, we get $k \geq \frac{n}{2c}$. Also, by $m' \leq (2 + c) \cdot n$, we get that $k \geq \frac{m'}{2c(2+c)}$.

Now, let $t = |\{(j, i) | (j, i) \in A \wedge 1 \leq i < j \leq n\}|$ be the number of arcs of H from a node of higher index to a node of lower index. We can assume that $t \geq c' \cdot k$.[15]

Now, consider the problem that a Separation Oracle for (5) has to solve — that is, consider Problem 2. Given a digraph H with m' arcs on the vertex set $[n']$, for each $1 \leq i < j \leq n'$, we set (i) $\Delta_{i,j} = \frac{1}{m'}$ if (i, j) is an arc of H, (ii) $\Delta_{i,j} = -\frac{1}{m'}$ if (j, i) is an arc of H, and (iii) $\Delta_{i,j} = 0$ otherwise. Now, let t be the number of pairs $1 \leq i < j \leq n'$ such that $\Delta_{i,j} < 0$. We set $D = \frac{t - c'k}{m'}$ — observe, then, that $D \geq 0$. The question is whether this assignment satisfies each constraint of (5), or whether it is off on some constraint by some additive constant $\alpha > 0$.

Observe that, for any permutation π of the vertices, if $C'(\pi)$ is the cost of π for the FAS instance H, then

$$C''(\pi) = \sum_{\substack{1 \leq i < j \leq n \\ i \prec_\pi j}} \Delta_{i,j} = \frac{C'(\pi)}{m'} - \frac{t}{m'}.$$

Observe that the total ℓ_1 weight of the $\overline{\Delta}$-tournament is 1; thus, the $|\Delta|_1 \leq 1$ constraint is satisfied. The reduction above shows that it is NP-hard to tell whether $\min_{\pi \in \mathbf{S}_n} C''(\pi) \leq \frac{k - t}{m'}$, or $\min_{\pi \in \mathbf{S}_n} C''(\pi) \geq \frac{c'k - t}{m'}$. Now, in the latter case, our choice of D guarantees that each constraint is satisfied; in the former case, instead, some constraint (in particular, a constraint corresponding to a permutation obtaining a minimum Feedback Arc Set) would be off by at least an additive $(c' - 1) \cdot \frac{k}{m'}$ term. Thus, it is NP-hard to determine if all the constraints of the separation oracle are satisfied, or if there exists a constraint that is off by an additive term of

$$\alpha = (c' - 1) \cdot \frac{k}{m'} \geq \frac{c' - 1}{2c^2 + 4c} = \Omega(1).$$

Thus, there exists a constant $\alpha > 0$ such that it is NP-hard to distinguish whether an assignment to (5) satisfies each constraint, or whether there exists some permutation constraint on which it is off by at least α. By the polynomial time equivalence of the separation oracle problem and the LP optimization problem (Grötschel et al., 1988), it is thus NP-hard to find the RUM that minimizes the maximum error with respect to a given tournament matrix. □

[12] That is, to the version of Problem 1, where each arc has the same weight of 1.

[13] Indeed, nodes of degree 0 can be removed from the graph without impacting the size of its optimal Vertex Covers. And, a graph of n nodes, having no nodes of degree 0, has to have at least $\frac{n}{2}$ edges.

[14] Recall that a graph G with m edges and maximum degree δ, cannot have a Vertex Cover of size smaller than $\frac{m}{\delta}$. That is, the Vertex Cover problem reduced to instances where $k < \frac{m}{\delta}$ is solvable in polynomial time. Thus, we can assume to reduce the set of hard Vertex Cover instances to those such that $k \geq \frac{m}{\delta}$.

[15] Given that we have a gap problem, which requires us to distinguish between graphs whose minimum FAS has value smaller than k, and graphs with minimum FAS having value larger than $c'k$, instances that do not satisfy this property can be easily solved in polynomial time — in such instances, the permutation $1 \prec 2 \prec \cdots \prec n$ has a FAS cost not larger than $t < c' \cdot k$.

C. Certifying optimality

If the RUM returned by Algorithm 1 is at distance 0 from the matrix P, its optimality is trivially guaranteed. If, instead, the distance is non-zero, we need to do extra work to certify it. One way to do this is to test whether the dual solution $\mathcal{D}$ returned by the last iteration of the loop of Algorithm 1 satisfies all dual constraints, i.e., we have to obtain the exact minimum FAS of the transitive tournament with $\mathcal{D}(\overline{\Delta})$ as the weights.

In the case of the `Jester` dataset in which $n = 100$, doing so is infeasible since the runtime of the exact dynamic programming algorithm for FAS is $O(n2^n)$; we then have no positive lower bound on the smallest distance to a RUM of that particular matrix. For the `SF` dataset in which $n = 35$, it so happens that the dual solution $\mathcal{D}$ obtained by the last iteration of Algorithm 1's loop induces a directed graph with only 28 vertices hit by at least one non-zero arc. Then, the minimum FAS instance is effectively reduced to a graph of order $n = 28$: the exact algorithm terminated in less than a minute, certifying (i) the feasibility of $\mathcal{D}$ for the unrestricted dual (3) and that (ii) the distance to the `SF` matrix of the RUM returned by Algorithm 1 is optimal (see Table 1).

Thus, while Algorithm 1 is a heuristic, it always returns a RUM and hence always gives a correct upper bound on the distance of the input matrix to RUMs. Moreover, with an extra final check on the feasibility for the dual of its last $\mathcal{D}$ solution, the heuristic can also certify a lower bound on the distance of the input matrix to the best approximating RUM. In our experiments, when applied, this final check was made with an exact algorithm. The Ellipsoid Algorithm, instead, makes this check using the Approximate Separation Oracle of Theorem 8.

D. The 2L model

The two-level (2L) model of Veerathu & Rajkumar (2021) partitions the n items into groups: each group is associated to a positive weight, and to a rank-2 tournament between its items. Pairwise matches between players in a same group are governed by the rank-2 tournament; if a player belongs to group i, and a second player belongs to group $j \neq i$, and if the weights of the two groups are w_i and w_j, then the first player wins with probability proportional to w_i (that is, with probability $\frac{w_i}{w_i+w_j}$).

One can easily show that there exist tournament matrices that can be represented via rank-2 tournaments (and thus via the 2L model), but not with RUMs. We show here that there also exist matrices that can be represented by RUMs, but not by 2L models.

In particular, consider the following tournament matrix over the set of players $X = \{(i, j) \mid 0 \leq i, j \leq 2\}$,

$$P_\alpha = \begin{array}{c} \begin{matrix} (0,0) & (0,1) & (0,2) & (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{matrix} \\ \begin{pmatrix} \cdot & \alpha & 1-\alpha & \alpha & \alpha & \alpha & 1-\alpha & 1-\alpha & 1-\alpha \\ 1-\alpha & \cdot & \alpha & \alpha & \alpha & \alpha & 1-\alpha & 1-\alpha & 1-\alpha \\ \alpha & 1-\alpha & \cdot & \alpha & \alpha & \alpha & 1-\alpha & 1-\alpha & 1-\alpha \\ 1-\alpha & 1-\alpha & 1-\alpha & \cdot & \alpha & 1-\alpha & \alpha & \alpha & \alpha \\ 1-\alpha & 1-\alpha & 1-\alpha & 1-\alpha & \cdot & \alpha & \alpha & \alpha & \alpha \\ 1-\alpha & 1-\alpha & 1-\alpha & \alpha & 1-\alpha & \cdot & \alpha & \alpha & \alpha \\ \alpha & \alpha & \alpha & 1-\alpha & 1-\alpha & 1-\alpha & \cdot & \alpha & 1-\alpha \\ \alpha & \alpha & \alpha & 1-\alpha & 1-\alpha & 1-\alpha & 1-\alpha & \cdot & \alpha \\ \alpha & \alpha & \alpha & 1-\alpha & 1-\alpha & 1-\alpha & \alpha & 1-\alpha & \cdot \end{pmatrix} \end{array} \begin{matrix} (0,0) \\ (0,1) \\ (0,2) \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{matrix}$$

for $\alpha \in [0, 1]$. With this matrix,

- player (i, j) beats player (i, k), for $k = (j + 1) \mod 3$, with probability $1 - \alpha$, and
- player (i, j) beats player (k, ℓ), for $k = (i + 1) \mod 3$, with probability $1 - \alpha$.

Note that for each $\alpha \neq \frac{1}{2}$, P_α has various disjoint non-transitive triplets, e.g., for each $i \in \{0, 1, 2\}$, the subset $\{(i, 0), (i, 1), (i, 2)\}$ is a non-transitive triplet. Moreover, P_α conjoins those three disjoint non-transitive triplets to form other non-transitive triplets: for each $i, j, k \in \{0, 1, 2\}$, the subset $\{(0, i), (1, j), (2, k)\}$ is also a non-transitive triplet.

We now show that (unrestricted) RUMs are able to represent such a P_α for each $\frac{1}{3} \leq \alpha \leq \frac{2}{3}$. We will later show that 2L models can represent P_α only if $\alpha = \frac{1}{2}$.

Lemma 13. *For each $\frac{1}{3} \leq \alpha \leq \frac{2}{3}$, the tournament matrix P_α* can *be represented with a RUM.*

Proof. We start by observing that if P_α is representable with a RUM, then $P_{1-\alpha}$ is also representable with a RUM. Indeed, for a permutation π of X, let $\overline{\pi}$ be the reverse of π. We also define the reverse of a RUM: the reverse $\overline{R}$ of RUM R is the RUM that assigns probability $R(\pi)$ to the permutation $\overline{\pi}$, for each π in the support of R. Then, for each pair a and b of items, the probability that a beats b in R is equal to the probability that b beats a in $\overline{R}$. Observe that if P_α requires the generic item a beat another item b with probability $p_{a,b}$, then $P_{1-\alpha}$ requires b to beat a with probability $p_{a,b}$. Thus, if R is a RUM for P_α, then $\overline{R}$ is a RUM for $P_{1-\alpha}$.

We now prove that the existence of a RUM for $P_{1/3}$ implies the existence of a RUM for P_α, for each $\frac{1}{3} \leq \alpha \leq \frac{2}{3}$.

First, if both the tournament matrix P' and the tournament matrix P'' admit RUMs, then for each $\beta \in [0,1]$, the tournament matrix $\beta \cdot P' + (1-\beta) \cdot P''$ also admits a RUM: to sample a permutation from this RUM, one first flips a coin with head probability β; if the coin comes up heads, one samples an independent permutation from the RUM of P', otherwise one samples an independent permutation from the rum of P''.

We have already proved that the existence of a RUM for $P_{1/3}$ implies the existence of a RUM for $P_{2/3}$. Let M_α be the RUM that is obtained by mixing the RUM for $P_{1/3}$, with weight $\beta = 2 - 3\alpha$, and the RUM for $P_{2/3}$, with weight $1 - \beta = 3\alpha - 1$. Then, an event that has probability $\frac{1}{3}$ in $P_{1/3}$ (and thus probability $\frac{2}{3}$ in $P_{2/3}$) ends up having probability $(2-3\alpha) \cdot \frac{1}{3} + (3\alpha - 1) \cdot \frac{2}{3} = \alpha$ in M_α; and, likewise, an event having probability $\frac{2}{3}$ in $P_{1/3}$ (and probability $\frac{1}{3}$ in $P_{2/3}$) ends up having probability $(2-3\alpha) \cdot \frac{2}{3} + (3\alpha - 1) \cdot \frac{1}{3} = 1 - \alpha$ in M_α.

Thus, the existence of a RUM for $P_{1/3}$ implies the existence of a RUM for P_α for each $\frac{1}{3} \leq \alpha \leq \frac{2}{3}$.

Then, there only remains to prove that there exists a RUM for $P_{1/3}$.

Consider the set $S = \{2 \prec 1 \prec 0, 0 \prec 2 \prec 1, 1 \prec 0 \prec 2\}$ of permutations. We will consider the uniform distribution on S; with this distribution, for each $i \in \{0,1,2\}$, the probability that i beats $(i+1) \mod 3$ is exactly $2/3$.

Our RUM $R_{1/3}$ behaves as follows. For each $i \in \{0,1,2\}$, let π_i be sampled independently and uniformly at random from S. Moreover, let π be also sampled independently and uniformly at random from S. Then, the RUM returns the permutation

$$\begin{pmatrix} (\pi(0), \pi_{\pi(0)}(0)) & \prec & (\pi(0), \pi_{\pi(0)}(1)) & \prec & (\pi(0), \pi_{\pi(0)}(2)) & \prec \\ (\pi(1), \pi_{\pi(1)}(0)) & \prec & (\pi(1), \pi_{\pi(1)}(1)) & \prec & (\pi(1), \pi_{\pi(1)}(2)) & \prec \\ (\pi(2), \pi_{\pi(2)}(0)) & \prec & (\pi(2), \pi_{\pi(2)}(1)) & \prec & (\pi(2), \pi_{\pi(2)}(2)) & \end{pmatrix}$$

With this RUM, for each player (i,j), and for $k = (j+1) \mod 3$, we have that $\Pr[(i,j) \text{ beats } (i,k)] = 2/3$.

Analogously, for each player (i,j), for $k = (i+1) \mod 3$, and for each $\ell \in \{0,1,2\}$, we that $\Pr[(i,j) \text{ beats } (k,\ell)] = 2/3$. Thus $R_{1/3}$ is a RUM that represents $P_{1/3}$. □

We then show that the 2L model of (Veerathu & Rajkumar, 2021) cannot represent P_α unless $\alpha = \frac{1}{2}$. In $P_{1/2}$, the winner of each single match is chosen by flipping a fair coin: thus, one can represent $P_{1/2}$ with a uniform MNL and, consequently, with a 2L model.

Our construction and our proof hinge on some constraints that 2L models have to obey when representing non-transitive triplets. As we will see, these constraints are incompatible with P_α for each $\alpha \neq \frac{1}{2}$.

Lemma 14. *For each $\alpha \neq \frac{1}{2}$, the tournament matrix P_α* cannot *be represented with a 2L model.*

Proof. Recall that $X = \{(i,j) \mid 0 \leq i,j \leq 2\}$ is the set of players; let us also define three disjoint subsets of X, that we call "parts", $X_i = \{(i,j) \mid 0 \leq j \leq 2\}$ for $0 \leq i \leq 2$.

Recall that the model of Veerathu & Rajkumar (2021) is on two levels. The model partitions the player set X in "groups" $G_0, G_1, \ldots$, and assigns a positive weight w_i to the generic group i. When two players, respectively from groups G_i and G_j, $i \neq j$, play against each other, a MNL is used to determine the winner: the player from group i wins with probability $\frac{w_i}{w_i + w_j}$, and the one from group j wins with probability $\frac{w_j}{w_i + w_j}$. When two players from the same group G_i play against each other, the game is decided by a rank-2 tournament.

Suppose by contradiction that a 2L model for representing P_α, $\alpha \neq \frac{1}{2}$ is composed of groups $G_0, \ldots, G_{t-1}$. Then,

- there cannot exist a part X_i that is split among 3 distinct groups, say, $G_{i_1}, G_{i_2}, G_{i_3}$: indeed, if this were the case, there would have to exist a permutation π of the indices such that $\frac{w_{i_{\pi(1)}}}{w_{i_{\pi(1)}}+w_{i_{\pi(2)}}} > \frac{1}{2}$, $\frac{w_{i_{\pi(2)}}}{w_{i_{\pi(2)}}+w_{i_{\pi(3)}}} > \frac{1}{2}$, $\frac{w_{i_{\pi(3)}}}{w_{i_{\pi(3)}}+w_{i_{\pi(1)}}} > \frac{1}{2}$, i.e., $w_{i_{\pi(1)}} > w_{i_{\pi(2)}} > w_{i_{\pi(3)}} > w_{i_{\pi(1)}}$, a contradiction;
- also, there cannot exist a part X_i that is split among 2 distinct groups, say, $(i,j),(i,k) \in G_a$ and $(i,\ell) \in G_b$ for $\{j,k,\ell\} = \{0,1,2\}$ and $a \neq b$. Indeed, if this were the case, the probability that the model assigns to (i,ℓ) beating (i,j), i.e., $\frac{w_b}{w_a+w_b}$, would be the same to that it assigns to (i,ℓ) beating (i,k), contradicting P_α's requirements.

Therefore, each X_i is to be fully contained in some group; in particular, there can either be 1, 2, or 3 groups.

Veerathu & Rajkumar (2021) prove that rank-2 tournaments are unable to capture tournament matrices P having indices a,b,c,d such that $P_{a,b}, P_{b,c}, P_{c,a} > \frac{1}{2}$ and $P_{a,d}, P_{b,d}, P_{c,d} > \frac{1}{2}$, or such that $P_{a,b}, P_{b,c}, P_{c,a} > \frac{1}{2}$ and $P_{a,d}, P_{b,d}, P_{c,d} < \frac{1}{2}$. In other words, rank-2 tournaments cannot represent a non-transitive triplet (a is stronger than b, b is stronger than c, and c is stronger than a) together with an extra item that is either stronger than each item in the triple, or weaker than each item in the triple (i.e., each of a,b,c is weaker than d, or each of a,b,c is stronger than d).

Thus, if the partition into groups of the 2L model of Veerathu & Rajkumar (2021) has a group G_i having some X_j as a subset, and containing some element of $X - X_j$, then the rank-2 tournament of G_i is unable to represent P_α restricted to G_i; thus the 2L model is unable to represent P_α.

It follows that a 2L model, to correctly represent P_α, has to contain exactly 3 groups, in fact, one group for each of the three "parts" X_0, X_1, X_2. Without loss of generality, let us assume that $G_i = X_i$ for $i \in \{0,1,2\}$.

Now, suppose that $\alpha < \frac{1}{2}$. To correctly represent P_α, the MNL weights would have to guarantee that the weight w_1 of G_1 is larger the weight w_2 of G_2 (since $P_{(0,0),(1,0)} = \alpha < \frac{1}{2}$), that $w_2 > w_3$ (since $P_{(1,0),(2,0)} = \alpha < \frac{1}{2}$), and that $w_3 > w_1$ (since $P_{(2,0),(0,0)} = \alpha < \frac{1}{2}$), which is a contradiction. Likewise, suppose that $\alpha > \frac{1}{2}$. In this case, to correctly represent P_α, the MNL weights would have to guarantee that $w_1 < w_2$ (since $P_{(0,0),(1,0)} = \alpha > \frac{1}{2}$), that $w_2 < w_3$ (since $P_{(1,0),(2,0)} = \alpha > \frac{1}{2}$), and that $w_3 < w_1$ (since $P_{(2,0),(0,0)} = \alpha > \frac{1}{2}$), which is also a contradiction.

Thus, the tournament matrix P_α cannot be represented by a 2L model, for each $\alpha \neq \frac{1}{2}$. □

The following is then immediate.

Corollary 15. *For each $\alpha \in \left[\frac{1}{3}, \frac{1}{2}\right) \cup \left(\frac{1}{2}, \frac{2}{3}\right]$, the tournament matrix P_α* can *be represented by a RUM and* cannot *be represented by 2L models.*

We point out that the same result could be obtained by a subset of the rows and columns of P_α—it is sufficient to have each element of one of the three parts X_i, together with an arbitrary element of a second part X_j, and an arbitrary element of the last part X_k, for $\{i,j,k\} = \{0,1,2\}$. We used all the 9 elements of the union of the three parts, since we felt that, this way, the RUM construction is clearer. In fact, while our construction only tensors an intransitive triplet with itself, RUMs can be shown to withstand arbitrarily many such tensorings, provided that each triplet's α is bounded between $\frac{1}{3}$ and $\frac{2}{3}$.