Aggregation of Partial Rankings, p-Ratings and Top-m Lists[*]

Nir Ailon[†]

Abstract

We study the problem of aggregating partial rankings. This problem is motivated by applications such as meta-searching and information retrieval, search engine spam fighting, e-commerce, learning from experts, analysis of population preference sampling, committee decision making and more. We improve recent constant factor approximation algorithms for aggregation of full rankings and generalize them to partial rankings. Our algorithms improved constant factor approximation with respect to all metrics discussed in Fagin et al's recent important work on comparing partial rankings. We pay special attention to two important types of partial rankings: the well-known top-m lists and the more general p-ratings which we define. We provide first evidence for hardness of aggregating them for constant m, p.

1 Introduction

Rank aggregation (see [4, 10, 14, 16, 17, 18, 19, 20, 21] and references therein) is the problem of finding a ranking (permutation) π of a ground set V of n elements combining information from a list $\pi_1, \ldots, \pi_k$ of input rankings (*votes*). This problem is motivated by many applications such as meta-searching and information retrieval, search engine spam fighting, e-commerce, learning from experts, analysis of population preference sampling, committee decision making and more. In addition to the practical motivation, there is a long history of theoretical interest in the mathematics arising from the problem (some classic milestones are [9, 11, 13, 25, 26]). For a nice survey, refer to [24].

Top-m rankings.

One of the main drawbacks of considering *full rankings* is that expecting full ranking information of V from all voters can be too much to ask for [16, 17, 18, 19, 20]. In the search engine example, it is unlikely that a search engine would provide a ranking of the entire set V of all web pages matching a given query. Instead, only the first $m \ll n$ top-ranked pages are returned. If we denote the full ranking (implicitly) computed by the search engine by $\pi : V \to \{1..n\}$ ($\pi(v)$ is the *rank* of v, smaller numbers meaning further "ahead" in the list), then the search engine returns only $\pi^{-1}(1), \ldots, \pi^{-1}(m)$ to the client. Lacking rank information among the elements $\pi^{-1}(m+1), \ldots, \pi^{-1}(n)$, the next best option is to assume they belong to one big *tie*, which is itself ranked in position $m+1$.

[*]Most of this work done while author was a student in the Department of Computer Science at Princeton University, and part while a member of the Institute for Advanced Study, supported by the National Science Foundation under agreement No. DMS-0111298. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

[†]Institute for Advanced Study, Princeton NJ

Partial rankings.

This last example together with others we present shortly motivate the consideration of *rankings with ties*, or, following more common terminology, *partial rankings*[1]. Note that there is nothing special about the number $m+1$ used in the example above except that it is strictly greater than m, the number we naturally (yet still arbitrarily) chose as the rank of the last page displayed by the search engine. A partial ranking can be abstracted as a mapping π from V to any totally ordered universe U, which we call the *rank universe*. What we call ties are simply *collisions* in the function. A full ranking is a collision-free mapping (an injection). The objective functions we define below will be independent of the actual choice of the rank universe. Our approach is *comparison* based (in contrast with *score* based aggregation which we do not consider here), and the rankings can be modeled as a list of comparisons between $\pi(u)$ and $\pi(v)$ for all unordered pairs (u, v) of elements in V. Using pairwise information has many advantages in problems related to both ranking and clustering [3, 22, 23, 28, 30].

Partial rankings arise naturally in many other problems. In many sports tournaments, ties are possible outcomes of single matches. In an election system, each voter can provide a partial ranking of the set V of candidates, where tying together a subset of candidates is a way of expressing neutrality with respect to that subset.

Ratings.

Another very important case of partial rankings are *ratings*. We define a p-rating to be a mapping from the ground set V to a rank universe U of size p for some fixed p (we may assume that $U = \{1, \ldots, p\}$). Some good examples are: (i) Each hotel in a set V of hotels is rated as $\star$, $\star\star$, $\star\star\star$, $\star\star\star\star$, or $\star\star\star\star\star$ by hotel critics ($p = 5$). (ii) Financial experts advise to either *sell,hold* or *buy* each stock in some set V ($p = 3$). (iii) A company identifies its strengths and weaknesses in customer service by distributing questionnaires to its clients. The questionnaire is a table with rows (V) corresponding to different customer service aspects, and columns (U) correspond to a range of p satisfaction levels. Each participating client marks an '$\times$' in a single box in each row, corresponding to their opinion[2].

Aggregation of partial rankings is hence a very natural and useful generalization of rank aggregation.

1.1 Choice of objective function

One of the challenges in this problem is to define a measure of disagreement (which we minimize) between the output and the input votes. The most natural question to ask before defining this measure is, what kind of object is the output σ? Should it be a member from the same space as $\pi_1, \ldots, \pi_k$? In the following example we argue against this approach. Assume $V = \{A, B, C\}$, and we wish to aggregate an input consisting of a list of ratings, where the rank universe is $U = \{good, bad\}$ (with $good < bad$). There are two voters, where the first rates A, B as *good* and C as *bad*, and the second rates A as *good* and B, C as *bad*. We write $\pi_1 = [AB, C]$ and $\pi_2 = [A, BC]$ as

[1]The term "partial ranking" used here should not be confused with two other standard objects: (1) *Partial order*, namely, a reflexive, transitive anti-symmetric binary relation; and (2) A ranking of a *subset* of V. In the search engine example, although only a subset of top m elements of V are returned, the remaining $n - m$ are implicitly assumed to be ranked behind.

[2]We assume no column is labeled "not applicable".

shorthand for this voting outcome. If we restrict the aggregation output to be a rating $\pi : V \to U$, then the two reasonable outputs are $\pi = [AB, C]$ and $\pi = [A, BC]$. The former misses the fact that in the greater scheme of things, A is better than B. The latter misses the fact that in the greater scheme of things, B is not worse than C. Also, the former solution cannot answer the question *"who is the best?"*, and the latter cannot answer the question *"who is the worst?"*. Intuitively, the output $[A, B, C]$ (i.e. a full ranking of V) seems to capture more pairwise information in this simple example. We therefore restrict the aggregation output in this work to be a *full* ranking of V (this approach was also assumed in [16]). We choose a measure of distance between partial rankings[3] generalizing the Kendall-τ measure [27] (originally defined as a *metric* on full rankings). The distance $d(\sigma, \pi)$ between partial rankings σ, π is the number of distinct $u, v \in V$ such that u is ranked strictly ahead of v in π and v is ranked strictly ahead of u in σ. Our goal is to minimize the sum of distances between the output and the individual partial rankings. This is the *Kemeny* approach to the aggregation problem and is considered to have many advantages [16, 17]. (See Section 7 for solution to a possible problem arising from top-m aggregation under this approach.)

Comparison with other choices: Fagin et al [18, 19, 20] provide a comprehensive picture on how to compare partial rankings In their work, they suggest different natural measures of distance between two partial rankings, extending the well known Kendall-τ and Spearman's footrule [15] metrics on full rankings. Their main contribution is in showing that all the extensions they study belong to a class $\mathcal{D}$ of metrics that are equivalent up to global constants, and hence by optimizing or approximating with respect to one we approximate with respect to all. The measure of distance d suggested in this abstract is, in fact, a special case of a family of generalizations they study. Alas, our choice of d is the black sheep in the family: it does not belong to class $\mathcal{D}$. In fact, it is not a proper distance function, because $d(\sigma, \pi)$ may be zero for two distinct σ, π. Luckily, in spite of the apparent difficulty in using an improper distance function, our results imply approximation algorithms[4] with respect to all metrics in class $\mathcal{D}$ as well! (this is shown in Section 6). Nevertheless, we argue that the apparent drawback in using d in many cases can be viewed as inherent to the problem itself and *not* to the choice of d. A voter tying together $u, v \in V$ may be incapable of providing comparison information due to infeasibility (as in the search engine top-m example) or coarseness of the rank universe (as in the p-rating example). Each voter implicitly has a full ranking $\hat{\pi}_i$, but provides us with only a projection $\pi_i = P_i(\hat{\pi}_i)$ onto a lower-dimensional space[5] S_i. What $d(\sigma, \pi_i)$ does is, in fact, measure the distance between the output full ranking σ and the preimage set $P_i^{-1}(\pi_i)$, that is, the distance to the closest full ranking in the preimage. The fact that the distance between a full-ranking and a partial-ranking may be zero is, in our view, analogous to the fact that two distinct points in Euclidean space may project onto the same point in a low-dimensional space: lack of information blurs distinction. The hope is that different voters project onto different subspaces, and hence, aggregation can recover a good consensus full ranking. (See Section 6 for further discussion.)

[3]By our assumption on the output, we care only about the distance between a full ranking and a partial ranking.
[4]Under our restriction of outputting full rankings only.
[5]Formally, the space of full rankings has $\binom{n}{2}$ coordinates encoding the order of all pairs using ± 1, and the subspace S_i fixes 0's in coordinates corresponding to tied pairs.

1.2 Our results and comparison with previous results

This work combines techniques from Ailon et al's recent work on full rank aggregation[6] [5] with Fagin et al's [18, 19] recent work on partial rankings. In addition to the methodological contribution (Sections 1.1 and 6), our main results are two approximation algorithms for aggregating partial rankings. The first (Section 3) is new a 2-approximation, generalizing a well-known [4] 2-approximation for full rank aggregation. We then present a new 3/2-approximation algorithm in Section 4, generalizing a recent algorithm [4] for full rank aggregation to the problem of partial rank aggregation. In addition to showing the applicability of the previous algorithm to the domain of partial rankings, we improve it using a new technique of perturbing the variables of an optimal solution to an LP relaxation before rounding it. In Section 6 we show that our algorithms also imply constant factor approximation with respect to the important class $\mathcal{D}$ of metrics discussed by Fagin et al in [18, 19]. They suggest a 3-approximation algorithm, but with respect to an objective function we do not use here.

Dwork et al successfully experiment with several heuristics in [16, 17], some based on Markov chains. They suggest a general scheme for aggregating partial rankings where a greedy post-processing step (called local Kemenization) is applied to the output of any algorithm. Local Kemenization is shown to have many nice properties, and it could easily be applied as a final step for our algorithms.

Finally (Section 5) we show that aggregation of partial rankings is in P when the rank universe U consists of 2 elements, but becomes NP-hard already when it consists of 3 elements (even for the special case of aggregation of top-2 lists). It was previously known that aggregation of partial rankings is NP-hard because rank aggregation is a special case (shown to be NP-Hard by Dwork et al [17]). Our result shows that the seemingly easier interesting cases of p-rating and top-m aggregation are already NP-Hard for extremely small rank universe (rank aggregation is *not* a special case of these problems). As far as we know, we are the first to define the problem of aggregating p-ratings (as a natural generalization of top-m lists) in the context of aggregating partial rankings and to provide evidence to both problems' hardness for constant m, p.

2 Definitions

We use $[n]$ to denote the integer set $\{1, \ldots, n\}$ equipped with the induced integer total order. We assume V is some ground set of n elements. Let E denote the set $\binom{V}{2}$, that is, the set of all unordered pairs $\{u, v\}$ of elements $u, v \in V$.

Definition 1 *A* partial-ranking *of V is a mapping $\pi : V \to U$ for some* rank universe *U equipped with a total order relation. Two partial-rankings $\pi : V \to U$, $\pi' : V \to U'$ of V are rank-equivalent if $\pi' = f \circ \pi$ for some strictly monotone[7] $f : \pi(V) \to \pi'(V)$ (hence $\pi = f^{-1} \circ \pi'$). A* full-ranking *of V is a partial-ranking that is an injection. A* strict *partial-ranking is a partial-ranking that is not a full-ranking. A* p-rating *of V is (up to rank-equivalence) a partial-ranking with range $[p]$. For $m \leq n$, a* top-m-ranking *of V is (up to rank-equivalence) an $(m+1)$-rating, where the preimages of $1, \ldots, m$ are singletons. The trivial partial-ranking $\mathbf{1}_V$ is the constant mapping.*

[6]By that we mean, aggregation of full (not partial) rankings.

[7]By $\pi(V)$ we mean the image of π.

(The definition of a p-rating was inspired by an example suggested by Moses Charikar.) For the sake of computation, we can always assume $U = [n]$. Given a partial-ranking $\pi : V \to U$, we write $u <_\pi v$ for $u, v \in V$ if $\pi(u) < \pi(v)$. We write $u =_\pi v$ if $\pi(u) = \pi(v)$. Similarly, we define $>_\pi, \leq_\pi$ and $\geq_\pi$.

Definition 2 *A shattering*[8] $\pi' * \pi$ *of* π *by* π' *is the unique (up to rank-equivalence) partial-ranking* σ *satisfying*

$$u <_\sigma v \iff u <_\pi v \text{ or } (u =_\pi v \text{ and } u <_{\pi'} v) .$$

The following facts are immediate to verify:

- The element $\mathbf{1}_V$ is the identity with respect to $*$,
- the shattering operator $*$ is associative, and,
- the shattering $\pi' * \pi$ can be computed in polynomial time (we omit the details of how to represent partial rankings as data structures and how to compute the shattering).

We now define a measure of distance between a full-ranking and a partial-ranking.

Definition 3 *Given two partial-rankings* σ *and* π *of* V*, the generalized Kendall-*τ *distance* $d(\sigma, \pi)$ *between the two is defined as the number of* $u, v \in V$ *such that* $u <_\sigma v$ *and* $v <_\pi u$*.*

Note that we will only need the distance between a full-ranking σ (the output) and a partial-ranking π (the votes). The distance between any partial-ranking and $\mathbf{1}_V$ is 0. Also note that d is not even a pseudometric, because $d([A,B],[AB]) = 0, d([B,A],[AB]) = 0$ and $d([A,B],[B,A]) = 1$, violating the triangle inequality. However, d restricted to full-rankings is a metric (in fact, an L^1 metric).

We are now ready to define the optimization problems considered in this work.

Definition 4 PartRankAgg *is the problem of, given a list* $\pi_1, \ldots, \pi_k$ *of partial-rankings of* V *(votes), outputting a full-ranking* π *minimizing* $\text{cost}(\pi) = \frac{1}{k}\sum_{i=1}^k d(\pi, \pi_i)$. RankAgg *is* PartRankAgg *with the restriction that the votes are full-rankings.* pRatingAgg *is* PartRankAgg *with the restriction that the votes are* p*-ratings.* Top$m$Agg *is* PartRankAgg *with the restriction that each vote* π_i *is a top-*m_i*-ranking for some* $m_i \leq m$*.*

Given an input $\pi_1, \ldots, \pi_k$ to PartRankAgg and distinct $u, v \in V$, we say that $u \equiv v$ if for all $i = 1, \ldots, k$, $u =_{\pi_i} v$. We define $w_{uv} = \frac{1}{k}|\{i : u <_{\pi_i} v\}|$. Clearly (i) $w_{uv} + w_{vu} \leq 1$ for all u, v; (ii) $u \equiv v \iff w_{uv} = w_{vu} = 0$; (iii) For any full-ranking π of V, $\text{cost}(\pi) = \sum_{u <_\pi v} w_{vu}$.

3 A 2-Approximation Algorithm for PartRankAgg

It is well known that RankAgg admits a very simple randomized 2-approximation algorithm (called pick-a-perm in [4]): simply output a choice of $\pi_1, \ldots, \pi_k$ uniformly at random. On expectation, such a choice has cost at most twice the optimal solution. One way of proving this is by arguing that d is a metric when restricted to the space of full-rankings. The 2-approximation argument

[8]In [18] the term *refinement* is used for the same operation. We follow their notation of $*$.

```
RepeatChoice ($V, \pi_1, \ldots, \pi_k$)

set $\pi \leftarrow \mathbf{1}_V$
set $\sigma \leftarrow$ arbitrary ranking of $V$
set $S \leftarrow \emptyset$
while $\exists u, v$ s.t. $u =_\pi v$ and $u \not\equiv v$
    choose $i \in [k] \setminus S$ uniformly at random
    set $\pi \leftarrow \pi_i * \pi$
    set $S \leftarrow S \cup \{i\}$
return $\sigma * \pi$
```

Figure 1: Pseudocode for RepeatChoice

easily follows from this fact. In our case, d is not a metric, and moreover, it is not clear how to turn some vote π_i (which could be a strict partial-ranking) into a full-ranking.

Our remedy to these problems is inspired by the idea that repeated play between two non-equivalent contenders will eventually be concluded (i.e. one will beat the other). Algorithm RepeatChoice (Figure 3) repeatedly chooses a random vote π_i (without repetitions) and shatters the current partial-ranking π until (almost) all ties are broken. We say "almost" because this scheme cannot break a tie between distinct u, v if $u \equiv v$. These ties are broken arbitrarily as the final step of the algorithm (equivalently, the result is shattered using an arbitrary full-ranking). If all of $\pi_1, \ldots, \pi_k$ are full-rankings (i.e. we are given input to RankAgg), then RepeatChoice is, in fact, equivalent to algorithm *pick-a-perm.*

Theorem 1 RepeatChoice *is a randomized expected* 2*-approximation algorithm for* PartRankAgg *with expected polynomial running time. The algorithm can be derandomized.*

Proof: The number of iterations is at most k and the running time is therefore clearly polynomial. We show the approximation guarantee. Fix two distinct $u, v \in V$. If $u \not\equiv v$, then it is obvious that $u <_\pi v$ with probability $w_{uv}/(w_{uv} + w_{vu})$ and $v <_\pi u$ with the remaining probability $w_{vu}/(w_{uv} + w_{vu})$. Hence, the total expected cost of the algorithm is

$$\begin{aligned} \mathbf{E}[\text{cost}(\pi)] &= \sum_{u \not\equiv v} \left(\frac{w_{uv}}{w_{uv} + w_{vu}} w_{vu} + \frac{w_{vu}}{w_{uv} + w_{vu}} w_{uv} \right) \\ &= \sum_{u \not\equiv v} \frac{2 w_{uv} w_{vu}}{w_{uv} + w_{vu}} \ . \end{aligned} \tag{1}$$

(Both summations are over *unordered* distinct pairs u, v). On the other hand, any optimal solution π^* satisfies $\text{cost}(\pi^*) \geq \sum_{u \not\equiv v} \min\{w_{uv}, w_{vu}\}$. Hence, $\mathbf{E}[cost(\pi)] \leq 2\,\text{cost}(\pi^*)$, as required.

We now show how to derandomize the choice of π_i for iteration t. Assume we chose $\pi_{i_1}, \ldots, \pi_{i_{t-1}}$ for distinct $i_1, \ldots, i_{t-1} \in [k]$ in the previous steps. Let $\pi^{(t-1)}$ denote the intermediate partial ranking computed after the $(t-1)$'th iteration of the main loop of RepeatChoice, namely, $\pi^{(t-1)} =$

$\pi_{i_{t-1}} * \cdots * \pi_{i_1} * \mathbf{1}_V$. Clearly,

$$\begin{aligned}\mathbf{E}[\mathrm{cost}(\pi) \mid i_1, \ldots, i_t] = &\sum_{i=1}^{k} d(\pi^{(t-1)}, \pi_k) \\ &+ \sum_{(u,v)\in P_<} w_{vu} \\ &+ \sum_{\{u,v\}\in P_=} 2\frac{w_{uv}w_{vu}}{w_{uv}+w_{vu}},\end{aligned}$$

whre $P_<$ denotes all ordered pairs (u, v) such that $u =_{\pi^{(t-1)}} v$ and $u <_{\pi_t} v$, and $P_=$ denotes all unordered nonequivalent pairs $\{u, v\}$ such that $u =_{\pi^{(t-1)}} v$ and $u =_{\pi_t} v$. Computing $\mathbf{E}[\mathrm{cost}(\pi) \mid i_1, \ldots, i_t]$ can be clearly done efficiently, and choosing i_t minimizing it at each step guarantees (by the principle of conditional expectations) a deterministic 2-approximation, as required. □

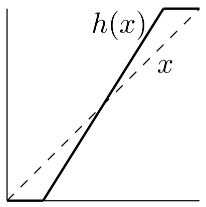
4 A (3/2)-Approximation Algorithm for PartRankAgg

The 2-approximation algorithm described in Section 3 took advantage of the fact that any optimal solution had to pay the minimum of w_{uv} and w_{vu} for any pair $u, v \in V$. In this section we take advantage of additional structure arising from considering triplets $u, v, y \in V$.

Fix three distinct elements $u, v, y \in V$. Clearly, $w_{uv} \leq w_{uy} + w_{yv}$, because any vote π_i that ranked u strictly before v must have either ranked u strictly before y or y strictly before v. This inequality is known as the triangle inequality on the weights induced by $\pi_1, \ldots, \pi_k$. In [4, 5], a (4/3)-approximation algorithm is presented for RankAgg. The (rather complicated) analysis relies heavily on the fact that the input is a list of full-rankings (and not just any partial-ranking). The algorithm there involves (1) solving an LP relaxation for RankAgg and using a randomized variant of the Quicksort algorithm for rounding it, (2) running pick-a-perm, and (3) outputting the better of the two results. In this section we consider the same LP and present a more complicated rounding technique, called LpKwikSort$_h$. This will result in the (3/2)-approximation algorithm for PartRankAgg. Our scheme does not involve taking the best of two algorithms, though it is quite possible that the best of LpKwikSort$_h$ and RepeatChoice gives a (4/3)-algorithm for PartRankAgg (we leave this as a conjecture for future work). Our new rounding technique LpKwikSort$_h$ is, as far as we know, the first algorithm that beats the 2-approximation for RankAgg by itself, without the need of running pick-a-perm and taking the best of the two. It is also the first algorithm that beats the 2-approximation for general minimum feedback arc-set in weighted tournaments with the triangle inequality [4].

To describe our improved algorithm, we first define a piecewise-linear function h mapping the real interval $[0, 1]$ onto itself. The function is defined as follows (see Section 7 for notes on the function):

$$h(x) = \begin{cases} 0 & 0 \leq x \leq \frac{1}{6} \\ \frac{3}{2}x - \frac{1}{4} & \frac{1}{6} < x \leq \frac{5}{6} \\ 1 & \frac{5}{6} < x \leq 1 \end{cases}$$



```
LpKwikSort$_h$ $(V, x = \{x_{uv}\}_{u,v \in V})$

if $V = \emptyset$ then
  return empty list
set $L \leftarrow \emptyset, R \leftarrow \emptyset$
pick pivot $v \in V$ uniformly at random
for all $u \in V$, $u \neq v$
  with probability $h(x_{uv})$
    add $u$ to $L$
  else (with remaining probability $1 - h(x_{uv}) = h(x_{vu})$)
    add $u$ to $R$
return concatenation of:
  LpKwikSort$_h$ $(L, x)$, $v$, LpKwikSort$_h$ $(R, x)$
```

Figure 2: Pseudocode for LpKwikSort$_h$

Note that for all $x \in [0,1]$, $h(x) = 1 - h(1-x)$ (in particular $h(1/2) = 1/2$). The function h will be used in a rounding algorithm of the following standard LP relaxation of PartRankAgg. The LP has a variable x_{uv} for each ordered $u \neq v$:

$$\begin{aligned} &\text{minimize} \sum_{u \neq v} (x_{uv} w_{vu} + x_{vu} w_{uv}) \text{ s.t.} \\ &x_{uv} + x_{vu} = 1 \ \forall u, v \\ &x_{uv} \leq x_{uy} + x_{yv} \ \forall u, v, y \\ &x_{uv} \geq 0 \ \forall u, v \end{aligned} \tag{2}$$

Clearly, if we could enforce $x_{uv} \in \{0,1\}$ for all u, v, we would have an exact IP for PartRankAgg. Given an optimal fractional solution $\{x_{uv}\}_{u,v}$, we round it using LpKwikSort$_h$ (Figure 4), which returns a list of all elements in V in some order. We convert this list into a full-ranking (e.g. a mapping onto $[n]$) in the obvious way (the first maps to 1, the second to 2, and so on). The algorithm improves the LP rounding algorithm in [4], by using h to bias probabilities of placing vertices on either side of the pivot vertex.

Theorem 2 LpKwikSort$_h$ *returns a full-ranking π with an expected cost of at most $3/2$ times the optimal LP value (and hence also the optimal cost of* PartRankAgg*).*

Note that this implies a bound of 3/2 on the LP integrality gap. The proof technique is very similar to [4, 5]. The main difficulty here is the use of the h-function and applicability to partial-rankings.

Proof: The basic idea is to decompose the costs into so-called *backward* costs (corresponding to triplets in V) and *forward* costs (corresponding to pairs in V). Let T denote the collection of all unordered triplets $\{u, v, y\}$ of three distinct vertices in V, and E denote the collection of all unordered pairs $\{u, v\}$ in V. For each $t = \{u, v, y\} \in T$, we define an event A_t in the random space induced by the execution of LpKwikSort$_h$. To define this event, we first notice that all elements of V are chosen as the pivot at some point in the recursive execution of LpKwikSort$_h$. We say that A_t occurred if when the first among u, v, y is chosen as pivot, the other two were input to the same recursive call. Note that conditioned on A_t, all three among u, v, y are equally likely to be that pivot (because the pivot is chosen uniformly at random). Let p_t denote $\Pr[A_t]$. Assume A_t occurs (and, say, v is the pivot). In that case, we will charge a *backward* cost to t corresponding to the random placement (i.e. L (left) or R (right)) of u, y. If u is placed in L and y in R, then the charge is w_{yu}, and if y is placed in L and u in R then the charge is w_{uy} (in all other cases, the charge is 0). It is immediate to verify that the total expected backward cost is $B = \sum_{t\in T} \frac{1}{3} p_t f(t)$, where for $t = \{u, v, y\}$,

$$\begin{aligned} f(t) = {} & h(x_{uv})h(x_{yu})w_{vy} + h(x_{vu})h(x_{uy})w_{yv} \\ & + h(x_{vy})h(x_{uv})w_{yu} + h(x_{yv})h(x_{vu})w_{uy} \\ & + h(x_{yu})h(x_{vy})w_{uv} + h(x_{uy})h(x_{yv})w_{vu} \ . \end{aligned} \tag{3}$$

For $e = \{u, v\} \in E$, let C_e denote the event that when the first among u, v was chosen as pivot, the other was input to the same recursive call. Let q_e denote $\Pr[C_e]$. Conditioned on C_e, the expected *forward* charge on e is $\hat{c}(e) = h(x_{uv})w_{vu} + h(x_{vu})w_{yv}$. The total expected forward cost is $F = \sum_{e\in E} q_e \hat{c}(e)$. The expected cost of the ranking returned by LpKwikSort$_h$ is $B + F$.

The LP value is $\sum_{e\in E} c(e)$, where $c(e) = (x_{uv}w_{vu} + x_{vu}w_{uv})$ for $e = \{u, v\}$. We decompose this as $B_{LP} + F_{LP}$, where $B_{LP} = \sum_{t\in T} \frac{1}{3} p_t g(t)$, for all $t = \{u, v, y\} \in T$:

$$\begin{aligned} g(t) = {} & (h(x_{uv})h(x_{yu}) + h(x_{vu})h(x_{uy}))c(\{vy\}) \\ & + (h(x_{vy})h(x_{uv}) + h(x_{yv})h(x_{vu}))c(\{yu\}) \\ & + (h(x_{yu})h(x_{vy}) + h(x_{uy})h(x_{yv}))c(\{uv\}) \ , \end{aligned} \tag{4}$$

and $F_{LP} = \sum_{e\in E} q_e c(e)$. It is not hard to verify that $B_{LP} + F_{LP}$ is indeed a decomposition of $\sum_{e\in E} c(e)$: each term $c(e)$ is accounted for exactly once (total) in B_{LP} and F_{LP}.

We want to show that $(B + F)/(B_{LP} + F_{LP}) \le 3/2$ whenever $B_{LP} + F_{LP} > 0$ and that if $B_{LP} + F_{LP} = 0$ then $B + F = 0$. To do so, it suffices to show that (1) for all $e \in E$, $\hat{c}(e)/c(e) \le 3/2$ whenever $c(e) > 0$ and $\hat{c}(e) = 0$ otherwise, and, (2) for all $t \in T$, $f(t)/g(t) \le 3/2$ whenever $g(t) > 0$ and $f(t) = 0$ otherwise.

To prove (1), it suffices to show (slightly changing notation) that for all $x, w_1, w_2 \in [0, 1]$ such that $w_1 + w_2 \le 1$, $a(x, w_1, w_2) = (h(x)w_1 + (1 - h(x))w_2)/(xw_1 + (1 - x)w_2) \le 3/2$ whenever the denominator is positive (it is clear that the numerator is 0 if the denominator is 0). We can therefore assume $w_1 + w_2 > 0$. Since we can divide w_1 and w_2 by $w_1 + w_2$ without changing the value of $a(x, w_1, w_2)$, we can assume $w_1 + w_2 = 1$. The function $b(x, w_1) = a(x, w_1, 1 - w_1)$ is a ratio of two linear functions in w_1 (for fixed x) and hence achieves its optima at $w_1 = 0$ or $w_1 = 1$. In the former case, $b(x, 0) = (1 - h(x))/(1 - x)$ and in the latter $b(x, 1) = h(x)/x$. By symmetry of h, it suffices to prove that $h(x)/x \le 3/2$ whenever $x \ne 0$, but clearly the maximum of $h(x)/x$ is obtained at $x = 5/6$, where $h(2/3)/(5/6) = 6/5 < 3/2$, as required.

To prove (2), Let $\mathbf{w} = (w_{uv}, w_{vu}, w_{vy}, w_{yv}, w_{yu}, w_{uy}) \in \mathbf{R}^6$ and $\mathbf{x} = (x_{uv}, x_{vu}, x_{vy}, x_{yv}, x_{yu}, x_{uy}) \in \mathbf{R}^6$. Let Δ_w denote the defining polytope of $\mathbf{w}$, carved by the triangle inequalities, non-negativity of the w's, and $w_{uv} + w_{vu} \le 1$, $w_{vy} + w_{yv} \le 1$, $w_{yu} + w_{uy} \le 1$. This polytope is also the convex closure of points $\mathbf{w}$ corresponding to individual votes on u, v, y. There are 13 such votes:

the 6 permutations $[u,v,y]$,$[u,y,v]$,$[v,u,y]$,$[v,y,u]$,$[y,u,v]$,$[y,v,u]$, the 6 votes tying 2 elements $[u,vy]$,$[v,yu]$,$[y,uv]$,$[uv,y]$,$[vy,u]$,$[yu,v]$ and the all-tie vote $[uvy]$. The corresponding points of Δ_w are (for example) $\mathbf{w} = (1,0,1,0,0,1)$ (corresponding to $[u,v,y]$), $\mathbf{w} = (1,0,0,0,0,1)$ (corresponding to $[u,vy]$) and $\mathbf{w} = (0,0,0,0,0,0)$ (corresponding to $[uvy]$). It is not hard to see that all the above 13 points are in fact vertices of Δ_w.

Assume $w_{uv} + w_{vu} + w_{vy} + w_{yv} + w_{yu} + w_{uy} > 0$ (otherwise $g(t) = f(t) = 0$). Let $c = \max\{w_{uv} + w_{vu}, w_{vy} + w_{yv}, w_{yu} + w_{uy}\} > 0$. By replacing $\mathbf{w}$ with $\mathbf{w}/c$ we do not change the value of $f(t)/g(t)$. Also, if $\mathbf{w} \in \Delta_w$ then so is $\mathbf{w}/c$. Hence, we may assume that either $w_{uv} + w_{vu} = 1$, $w_{vy} + w_{yv} = 1$ or $w_{yu} + w_{uy} = 1$. Without loss of generality (and due to symmetry), assume from now on that $w_{uv} + w_{vu} = 1$. Let H denote the hyperplane in $\mathbf{R}^6$ defined by this constraint. We restrict our attention to $H \cap \Delta_w$.

To simplify the proof, instead of working with the ratio $f(t)/g(t)$, we can equivalently show that $z(t) := f(t) - \frac{3}{2}g(t) \le 0$ for all $t \in T$. The function $z(t)$ is linear in w_{uv},w_{vu},w_{vy},w_{yv},w_{yu} and w_{uy} (assuming fixed $\mathbf{x}$) and hence achieves its optima on vertices of $H \cap \Delta_w$. Since H is a defining hyperplane for Δ_w (the entire polytope is contained in one side of the hyperplane), the vertices of $H \cap \Delta_w$ are exactly the intersection of the set of vertices of Δ_w with H. There are 10 such vertices, corresponding to the 6 permutations and $[u,yv]$,$[yv,u]$,$[v,uy]$,$[uy,v]$. Substituting each of the 10 vertices for $\mathbf{w}$ in $z(t)$, we obtain 10 functions in $\mathbf{x}$. These 10 functions fall into 2 symmetry classes, one corresponding to a permutation (say, $\mathbf{w} = (1,0,1,0,0,1)$) and the other to a 2-tie vote (say, $\mathbf{w} = (1,0,0,0,0,1)$). We therefore consider two functions, corresponding to the two substitutions for $\mathbf{w}$. Substituting the two $\mathbf{w}$'s into z, the two corresponding functions $r(\mathbf{x})$ and $s(\mathbf{x})$ are

$$\begin{aligned}
&r(\mathbf{x}) = \\
&\quad\Big(h(x_{uv})h(x_{yu}) + h(x_{yv})h(x_{vu}) + h(x_{yu})h(x_{vy})\Big) \\
&\quad - \frac{3}{2}\Big(\big(h(x_{uv})h(x_{yu}) + h(x_{vu})h(x_{uy})\big)x_{yv} \\
&\quad + \big(h(x_{vy})h(x_{uv}) + h(x_{yv})h(x_{vu})\big)x_{yu} \\
&\quad + \big(h(x_{yu})h(x_{vy}) + h(x_{uy})h(x_{yv})\big)x_{vu}\Big) \\
&s(\mathbf{x}) = \\
&\quad\Big(h(x_{yv})h(x_{vu}) + h(x_{yu})h(x_{vy})\Big) \\
&\quad - \frac{3}{2}\Big(\big(h(x_{vy})h(x_{uv}) + h(x_{yv})h(x_{vu})\big)x_{yu} \\
&\quad + \big(h(x_{yu})h(x_{vy}) + h(x_{uy})h(x_{yv})\big)x_{vu}\Big) \,.
\end{aligned} \tag{5}$$

We may substitute $1 - x_{uv}$ for x_{vu}, $1 - x_{vy}$ for x_{yv} and $1 - x_{yu}$ for x_{uy} (from the LP constraints). Abusing notation, we keep using s, r to denote the functions after the substitution, and $\mathbf{x} \in \mathbf{R}^3$ to denote (x_{uv}, x_{vy}, x_{yu}). By the triangle inequality, we have $1 \le x_{uv} + x_{vy} + x_{yu} \le 2$. Let $\Delta_x \subseteq \mathbf{R}^3$ denote the polytope of possible $\mathbf{x}$'s. The function h is linear on each one of the intervals $I_1 = [0, 1/6]$, $I_2 = [1/6, 5/6]$ and $I_3 = [5/6, 1]$, and hence it will be useful to separately analyze s and r on each of the 27 domains $D_{i,j,k} = \Delta_x \cap (I_i \times I_j \times I_k)$ for $i, j, k \in \{1, 2, 3\}$. The remaining of the proof is elementary case-by case analysis of the multinomials s and r on the 27 corresponding domains. In Tables 1 and 2 we present the functions s and r (respectively) after substitution on these domains. For ease of notation in the table, we use x_1, x_2 and x_3 instead of x_{uv}, x_{vy} and x_{yu}, respectively. To simplify the verification of the tables, note that the restriction of s and r to $D_{i,j,k}$ for all i, j, k is *trilinear* (linear in each of the three variables x_1, x_2, x_3 when the other two are fixed). Therefore, any face[9] of the polytope $D_{i,j,k}$ parallel to an axis can be removed from consideration when studying the the maxima of s, r on $D_{i,j,k}$ (because such faces are unions of axis-parallel line segments on which s, r are linear; the endpoints of these line segments are contained in lower-dimensional faces). It remains to consider only the 2, 1 and 0-dimensional faces of $D_{i,j,k}$ parallel to the hyperplanes defined by $x_1 + x_2 + x_3 = 2$ and $x_1 + x_2 + x_3 = 1$.

One way to automate the tedious computation of the maxima of s and r is to use the cylindrical algebraic decomposition algorithm [7, 8, 12]. The polynomials in question are simple enough to make the computation feasible. Indeed we did this using Mathematica software, and the maximum outputted by the program was always ≤ 0, as required. The program and its output can be found on the author's website [1]. A complete analysis (not computer aided) will be included in the full version on the author's website.

□

5 2RatingAgg is in P, Top2Agg is NP-Hard

Theorem 3 *pRatingAgg has a polynomial-time algorithm when $p = 2$ (hence, so does Top$m$Agg for $m = 1$).*

Proof: Given an instance $\pi_1, \ldots, \pi_k : V \to [2]$ of 2RatingAgg, for each $v \in V$ we let n_v be the number integers $i \in [k]$ such that $\pi_i(v) = 2$. It is easy to see that $n_u > n_v \iff w_{vu} > w_{uv}$. Therefore, if for some full-ranking $\pi : V \to [n]$ there are $u, v \in V$ such that $\pi(u) + 1 = \pi(v)$ and $n_u > n_v$, then the cost of π will strictly improve if we swap the values of π at u and v. Hence sorting V in increasing n_v order (breaking ties arbitrarily) are exactly the optimal rankings. Such a ranking can be computed in polynomial time. □

Theorem 4 *Top$m$Agg is NP-Hard when $m = 2$ (hence, so is pRatingAgg for $p = 3$).*

Proof: We show a reduction from minimum feedback arc-set in tournaments (MinFasTour), which was recently shown to NP-Hard by Noga Alon [6] (based on a derandomization of a reduction from [4]). MinFasTour is the problem of, given a tournament $T = (V, A)$, finding a ranking π of V minimizing the number of backward edges, namely $\text{cost}_T(\pi) = \sum_{u <_\pi v} \mathbf{1}_{(v,u) \in A}$, where $\mathbf{1}_P$ is 1 if predicate P is true and 0 otherwise.

Given an instance $T = (V, A)$ of MinFasTour, we define a corresponding instance of Top2Agg. The votes $\pi_{\{u,v\}} : V \to [3]$ are indexed using all $\binom{n}{2}$ unordered pairs $\{u, v\}$, where for each $\{u, v\}$ and $y \neq u, v$:

$$\pi_{\{u,v\}}(u) = \begin{cases} 1 & (u, v) \in A \\ 2 & (v, u) \in A \end{cases}$$

$$\pi_{\{u,v\}}(y) = 3 \ .$$

[9]Faces of polytopes are *open* sets (in their affine closure) by convention.

i, j, k	$s\|_{D_{i,j,k}}$	$\max s$	$\operatorname{argmax} s$
$1,1,2$	$(-7+6x_3-3x_1(-5+6x_3))/8$	$-1/4$	$(\frac{5}{64},\frac{1}{8},\frac{5}{6})$
$1,1,3$	$1-3x_3/2$	$-1/4$	$(\frac{1}{6},0,\frac{5}{6})$
$1,2,1$	$-(-5+6x_2)(-1+3x_1-3x_3)/8$	0	$(\frac{5}{64},\frac{5}{6},\frac{1}{8})$
$1,2,2$	$3(-6+8x_2+6x_3-12x_2x_3+ x_1(13-18x_3+18x_2(-1+2x_3))))/16$	0	$(\frac{5}{64},\frac{5}{6},\frac{1}{6})$
$1,2,3$	$(11+3x_1(-1+6x_2)+18x_2(-1+x_3)-15x_3)/8$	$-1/4$	$(\frac{1}{6},\frac{1}{4},\frac{5}{6})$
$1,3,1$	0	0	const. func.
$1,3,2$	$(-1+3x_1)(-1+6x_3)/8$	0	$(\frac{5}{64},\frac{59}{64},\frac{1}{6})$
$1,3,3$	$(-1+3x_1)/2$	$-1/4$	$(\frac{1}{6},\frac{5}{6},\frac{5}{6})$
$2,1,1$	$(-2+3(-5+6x_1)x_3)/8$	$-1/4$	$(\frac{5}{6},\frac{1}{8},\frac{5}{64})$
$2,1,2$	$(-5+3x_1+3x_3)/8$	0	$(\frac{5}{6},\frac{5}{64},\frac{5}{6})$
$2,1,3$	$(-5+6x_1)(-2+3x_3)/8$	0	$(\frac{5}{6},\frac{5}{64},\frac{59}{64})$
$2,2,1$	$(-5+(-39+54x_1)x_3+x_2(6-54(-1+2x_1)x_3))/16$	0	$(\frac{1}{6},\frac{5}{6},\frac{3}{128})$
$2,2,2$	$(-13+x_1(9-18x_2)-18x_2(-1+x_3)+9x_3)/16$	0	$(\frac{5}{6},\frac{1}{6},\frac{5}{6})$
$2,2,3$	$-3(-9+13x_3+6x_1(-1+2x_2)(-2+3x_3)- 2x_2(-7+x_3))/16$	0	$(\frac{5}{6},\frac{1}{6},\frac{59}{64})$
$2,3,1$	$-3(-1+6x_1)x_3/8$	0	$(\frac{1}{6},\frac{59}{64},\frac{5}{64})$
$2,3,2$	$(1-3x_1-3x_3)/8$	0	$(\frac{1}{6},\frac{59}{64},\frac{1}{6})$
$2,3,3$	$(-4+3x_3-6x_1(-2+3x_3))/8$	$-1/4$	$(\frac{1}{6},\frac{7}{8},\frac{59}{64})$
$3,1,1$	$3(-1+x_1)/2$	0	$(1,0,0)$
$3,1,2$	$-3(-1+x_1)(-5+6x_3)/8$	0	$(1,\frac{5}{64},\frac{1}{2})$
$3,1,3$	0	0	const. func.
$3,2,1$	$-3(5+x_1(-5+6x_2)+6x_2(-1+x_3)-x_3)/8$	0	$(1,\frac{9}{32},0)$
$3,2,2$	$(-38+54x_3-12x_2(-4+9x_3)+ 3x_1(13-18x_3+18x_2(-1+2x_3))))/16$	0	$(1,\frac{1}{6},\frac{1}{2})$
$3,2,3$	$(-1+6x_2)(-1+3x_1-3x_3)/8$	0	$(\frac{59}{64},\frac{1}{6},\frac{7}{8})$
$3,3,1$	$-3x_3/2$	0	$(\frac{5}{6},\frac{5}{6},0)$
$3,3,2$	$(1-18x_3+3x_1(-1+6x_3))/8$	$-1/4$	$(\frac{59}{64},\frac{7}{8},\frac{1}{6})$

Table 1: Analysis of s on 27 domains. The entries $(i,j,k)=(1,1,1)$ and $(i,j,k)=(3,3,3)$ do not appear as $D_{i,j,k}=\emptyset$ there.

i,j,k	$r\|_{D_{i,j,k}}$	$\max r$	$\operatorname{argmax} r$
$1,1,2$	$(-22+24x_3-3x_1(-5+6x_3)-3x_2(-5+6x_3))/8$	$-1/4$	$(\frac{5}{64},\frac{1}{8},\frac{5}{6})$
$1,1,3$	$1-3x_3/2$	$-1/4$	$(\frac{1}{6},0,\frac{5}{6})$
$1,2,1$	$(-17-3x_1(-5+6x_2)-15x_3+18x_2(1+x_3))/8$	$-1/4$	$(\frac{5}{64},\frac{5}{6},\frac{1}{8})$
$1,2,2$	$3(2(-8+x_2(9-12x_3)+9x_3)+$ $x_1(13-18x_3+18x_2(-1+2x_3))))/16$	$-1/4$	$(\frac{1}{6},\frac{11}{64},\frac{5}{6})$
$1,2,3$	$(11+3x_1(-1+6x_2)+18x_2(-1+x_3)-15x_3)/8$	$-1/4$	$(\frac{1}{6},\frac{1}{4},\frac{5}{6})$
$1,3,1$	$3(-1+x_2)/2$	0	$(0,1,0)$
$1,3,2$	$(-14+15x_2+12x_3-18x_2x_3+3x_1(-1+6x_3))/8$	0	$(\frac{5}{64},1,\frac{1}{6})$
$1,3,3$	$(-1+3x_1)/2$	$-1/4$	$(\frac{1}{6},\frac{5}{6},\frac{5}{6})$
$2,1,1$	$(-17+15x_2-18x_1(-1+x_2-x_3)-15x_3)/8$	$-1/4$	$(\frac{5}{6},\frac{5}{64},\frac{1}{8})$
$2,1,2$	$3(-16+13x_2+18x_3-18x_2x_3+$ $6x_1(3-4x_3+x_2(-3+6x_3))))/16$	$-1/4$	$(\frac{5}{6},\frac{5}{64},\frac{1}{6})$
$2,1,3$	$(11-3x_2-15x_3+18x_1(-1+x_2+x_3))/8$	$-1/4$	$(\frac{1}{6},\frac{5}{64},\frac{5}{6})$
$2,2,1$	$(-35-39x_3+18x_2(2+3x_3)-$ $18x_1(-2-3x_3+x_2(2+6x_3))))/16$	0	$(\frac{5}{6},\frac{5}{6},0)$
$2,2,2$	$3(-17+x_2(19-24x_3)+19x_3+$ $x_1(19-24x_3+12x_2(-2+3x_3))))/16$	$-7/128$	$(\frac{5}{6},\frac{7}{12},\frac{7}{12})$
$2,2,3$	$(29-6x_1(8+18x_2(-1+x_3)-9x_3)-$ $39x_3+6x_2(-8+9x_3))/16$	$-7/128$	$(\frac{7}{12},\frac{7}{12},\frac{5}{6})$
$2,3,1$	$-3(5-5x_2-x_3+6x_1(-1+x_2+x_3))/8$	0	$(\frac{5}{6},\frac{57}{64},0)$
$2,3,2$	$3(-12+13x_2+14x_3-18x_2x_3+$ $2x_1(7-12x_3+9x_2(-1+2x_3))))/16$	0	$(\frac{1}{6},1,\frac{1}{6})$
$2,3,3$	$3(-1-x_2+x_1(2+6x_2-6x_3)+x_3)/8$	$-1/8$	$(\frac{1}{3},\frac{5}{6},\frac{5}{6})$
$3,1,1$	$3(-1+x_1)/2$	0	$(1,0,0)$
$3,1,2$	$(-14-3x_2+12x_3+18x_2x_3-3x_1(-5+6x_3))/8$	0	$(1,\frac{5}{64},\frac{1}{6})$
$3,1,3$	$(-1+3x_2)/2$	$-1/4$	$(\frac{5}{6},\frac{1}{6},\frac{5}{6})$
$3,2,1$	$-3(5+x_1(-5+6x_2)+6x_2(-1+x_3)-x_3)/8$	0	$(1,\frac{9}{32},0)$
$3,2,2$	$3(2(-6+x_2(7-12x_3)+7x_3)+$ $x_1(13-18x_3+18x_2(-1+2x_3))))/16$	0	$(1,\frac{1}{6},\frac{1}{6})$
$3,2,3$	$3(-1+x_1(-1+6x_2)+x_2(2-6x_3)+x_3)/8$	$-1/8$	$(\frac{5}{6},\frac{1}{3},\frac{5}{6})$
$3,3,1$	$-3x_3/2$	0	$(\frac{5}{6},\frac{5}{6},0)$
$3,3,2$	$(2-24x_3+3x_1(-1+6x_3)+3x_2(-1+6x_3))/8$	$-1/8$	$(\frac{5}{6},\frac{5}{6},\frac{1}{3})$

Table 2: Analysis of r on 27 domains. The entries $(i,j,k)=(1,1,1)$ and $(i,j,k)=(3,3,3)$ do not appear as $D_{i,j,k}=\emptyset$ there.

It is not hard to see that for this instance, any full-ranking π of V has cost

$$\mathrm{cost}(\pi) = \frac{1}{\binom{n}{2}} \sum_{u <_\pi v} \left(\mathbf{1}_{(v,u)\in A} + (n-2) \right) .$$

(The $n-2$ term is from the contribution of $\pi_{\{v,y\}}$ for $y \neq u, v$.) Therefore, $\mathrm{cost}(\pi)$ and $\mathrm{cost}_T(\pi)$ are linearly related. This completes the NP-Hardness reduction. □

6 Applicability to other metrics on partial-rankings

As mentioned in Section 1.1, d belongs to a family of distance functions studied by Fagin et al [18]. This family is parametrized by a real number $0 \leq \rho \leq 1$. Our distance function d is obtained by taking $\rho = 0$ and is not a proper distance function[10] because the distance between two partial-rankings may be zero (in fact, even if one of the arguments is a full-ranking). The distance $d_\rho(\sigma, \pi)$ between two partial-rankings σ, π is defined in [18] as $d(\sigma, \pi) + \rho d'(\sigma, \pi)$, where $d'(\sigma, \pi)$ is the number of unordered pairs $\{u, v\}$ such that either $u =_\pi v$ and $u \neq_\sigma v$, or $u =_\sigma v$ and $u \neq_\pi v$. The function $d'(\sigma, \pi)$ measures the difference between the two clusterings of V induced by the tie-relations in σ and π (in fact, it is exactly the *consensus-clustering* metric [22, 29, 30]). It is shown in [18] that $d_{1/2}$ is a metric belonging to an important class $\mathcal{D}$ of many other equivalent metrics derived from both the Kendall-τ and the Spearman's footrule metrics on full-rankings. We will not go into the definitions of the metrics belonging to $\mathcal{D}$ studied in [18].

If we used $d_{1/2}$ instead of d in the definition of our objective function, this would add a constant depending on the input $\pi_1, \ldots, \pi_k$ (and not on the output). Indeed, since an output σ is a full-ranking, $d'(\sigma, \pi_i)$ is simply the number of pairs u, v that are tied in π_i. Therefore, by Theorems 1 and 2, algorithms RepeatChoice and LpKwikSort$_h$ are respectively 2 and 3/2 factor approximation algorithms with respect to $d_{1/2}$. By up-to-constant equivalence of all the metrics in $\mathcal{D}$, this also implies constant-factor approximations with respect to them as well.

By the discussion in Section 1.1, partial-rankings can be thought of as *partially revealed* full-rankings. Assuming this interpretation, the metrics in class $\mathcal{D}$ can be thought of as methods for compensating for unrevealed information. We suggest the following alternative approach for compensating. For a partial-ranking π, let $\mu(\pi)$ count the number of pairs u, v that are *not* tied in π (for full-rankings, this is $\binom{n}{2}$). Intuitively, μ measures the amount of information revealed by π. For a full-ranking σ and partial-ranking π, define $\hat{d}(\sigma, \pi)$ as $d(\sigma, \pi)\binom{n}{2}/\mu(\pi)$ (if $\mu(\pi) = 0$ then also $d(\sigma, \pi) = 0$ and we define $\hat{d}(\sigma, \pi) = 0$). Intuitively, this spreads the available distance information evenly across the missing information. We suggest using $\hat{d}$ as an alternative to using d and to other distance measures that are used and studied in the literature. Fortunately, algorithms RepeatChoice and LpKwikSort$_h$ generalize to this normalized version, as normalization is achieved by attaching an "importance" weight of $\binom{n}{2}/\mu(\pi_i)$ to each voter. For the sake of simplicity, we omit the simple generalization to weighted voters (it is easy to see, for example, that an integer weight of ω assigned to a voter can be simulated by considering ω unweighted copies of the voter). Also note that the hardness statement in Theorem 4 applies to the normalized version as well, because the hard instances in the proof would assign the same normalization weight to all voters.

[10]In [18] they use the term "distance function" for what we call "proper distance function", and aside from the definition, they don't deal much with $\rho = 0$.

7 Concluding Remarks

1. Does there exist a PTAS for RankAgg? for pRatingAgg?

2. Is Top$m$Agg NP-Hard for $k = o(n)$ voters? (RankAgg is NP-Hard for 4 voters [16].)

3. Is taking the best of LpKwikSort$_h$ and RepeatChoice better than a 4/3-approximation for PartRankAgg (as the results in [4] would suggest)?

4. One may argue that the algorithms in this paper hardly apply to Top$m$Agg, because (as the search engine example suggests) n may be too large to work with. However, it is trivial to show that we can run our algorithms considering only the collection of elements of V topping at least one voter, and (implicitly) place the rest as the output tail (their internal order has no effect on the cost). (In [16] they make the practical assumption of the input not containing elements topping no voter in the first place.)

5. The choice of the h-function in Section 4 is optimal in the sense that using any other h-function with the same rounding algorithm and the same analysis technique cannot result in a better than 3/2 approximation (though different analysis might lead to a better result). Indeed, for $\mathbf{w} = (1, 0, 1, 0, 0, 1)$ and $\mathbf{x} = (1/2, 1/2, 1, 0, 1/2, 1/2)$ we get $f(t)/g(t) = 3/2$, and one can show easily that h must satisfy $h(0) = 0, h(1/2) = 1/2, h(1) = 1$. In order to come up with h, we first found the unique symmetric function $\hat{h}$ with the property that $f(t)/g(t) = 3/2$ for $\mathbf{w} = (1, 0, 1, 0, 0, 1)$ and $\mathbf{x} = (1 - \alpha, \alpha, 1, 0, \alpha, 1 - \alpha)$, for all $0 < \alpha \leq 1/2$. This function is $\hat{h}(x) = 1 - \sqrt{1 - \frac{3}{2}x}$ for $x \leq 1/2$ and its symmetric completion for $x > 1/2$. Then h is a piecewise linear approximation of $\hat{h}$.Working with $\hat{h}$ seems more difficult for analysis though may prove to be good in practice. Note that in an earlier version [2] a different, slightly more complicated h was used, but it was subsequently discovered that the one used here suffices.

6. It would be nice to somehow generalized Coppersmith et al's [14] recent important result relating the Borda score-based rank aggregation algorithm with the Kemeny-optimal (for full-rankings). Can this be done for partial-rankings as well?

Acknowledgements

We would like to thank Moses Charikar, Alantha Newman, Ron Fagin, Cynthia Dwork, Kunal Talwar, Steve Chien, Seffi Naor, Joel Predd and Amit Agarwal for enlightening discussions and feedback.

References

[1] N. Ailon. A mathematica program for automatically proving inequalities from 'Aggregation of partial rankings, p-ratings and top-m lists'. *http://www.math.ias.edu/˜nailon/proof.nb.*

[2] N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[3] N. Ailon and M. Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[4] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 684–693, 2005.

[5] N. Ailon, M. Charikar, and A. Newman. Proofs of conjectures in 'aggregating inconsistent information: ranking and clustering'. *Technical Report, Princeton University*, TR-719-05, 2005.

[6] N. Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, to appear.

[7] D. Arnon, G. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM J. on Computing*, 13, 1984.

[8] D. Arnon, G. Collins, and S. McCallum. Cylindrical algebraic decomposition II: The adjacency algorithm for the plane. *SIAM J. on Computing*, 13, 1984.

[9] K. J. Arrow. *Social Choice and Individual Values.* John Wiley and Sons, New York, 1951.

[10] J. Aslam and M. Montague. Condorcet fusion for improved retrieval. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 538–548, 2002.

[11] J. C. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781.

[12] G. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of the 2nd GI Conference on Automate Theory and Formal Languages*, Vol. 33 of Lecture Notes in Computer Science, pages 134–183. Springer-Verlag, Berlin, 1975.

[13] M.-J. Condorcet. Éssai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. 1785.

[14] D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournamnets. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[15] P. Diaconis and R. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society, Series B*, 39(2):262–268, 1977.

[16] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International Conference on the World Wide Web (WWW10)*, pages 613–622, Hong Kong, 2001.

[17] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation revisited. *Manuscript*, 2001.

[18] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing partial rankings. *SIAM Journal of Discrete Mathematica.*

[19] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 47–58, 2004.

[20] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 28–36, Baltimore, 2003.

[21] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 301–312, San Diego, 2003.

[22] V. Filkov and S. Skiena. Integrating microarray data by consensus clustering. In *Proceedings of International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 418–425, Sacramento, 2003.

[23] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, Tokyo, 2005. To appear.

[24] J. Hodge and R.E.Klima. *The Mathematics of Voting and Elections: A Hands-On Approach*, volume 22 of *Mathematical World.* AMS, 2000.

[25] J. Kemeny and J. Snell. *Mathematical Models in the Social Sciences.* Blaisdell, New York, 1962. Reprinted by MIT Press, Cambridge, 1972.

[26] J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959.

[27] M. Kendall and J. D. Gibbons. *Rank correlation methods.* Edward Arnold, London, 1990.

[28] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[29] A. Strehl. Relationship-based clustering and cluster ensembles for high-dimensional data mining. *PhD Dissertation, University of Texas at Austin*, May 2002.

[30] Y. Wakabayashi. The complexity of computing medians of relations. *Resenhas*, 3(3):323–349, 1998.