Deterministic Algorithms for Rank Aggregation and Other Ranking and Clustering Problems

Anke van Zuylen* and David P. Williamson**

School of Operations Research and Information Engineering,
Cornell University, Ithaca, NY 14853.
Fax: (607) 255-9129.
avz2@cornell.edu
dpw@cs.cornell.edu

Abstract. We consider ranking and clustering problems related to the aggregation of inconsistent information. Ailon, Charikar, and Newman [1] proposed randomized constant factor approximation algorithms for these problems. Together with Hegde and Jain, we recently proposed deterministic versions of some of these randomized algorithms [2]. With one exception, these algorithms required the solution of a linear programming relaxation. In this paper, we introduce a purely combinatorial deterministic pivoting algorithm for weighted ranking problems with weights that satisfy the triangle inequality; our analysis is quite simple. We then shown how to use this algorithm to get the first deterministic combinatorial approximation algorithm for the partial rank aggregation problem with performance guarantee better than 2. In addition, we extend our approach to the linear programming based algorithms in Ailon et al. [1] and Ailon [3]. Finally, we show that constrained rank aggregation is not harder than unconstrained rank aggregation.

Keywords: derandomization, rank aggregation, feedback arc set in tournaments

1 Introduction

We consider the problem of ranking or clustering a set of elements, based on input information for each pair of elements. The objective is to find a solution that minimizes the deviation from the input information. For example, we may want to cluster webpages based on similarity scores, where for each pair of pages we have a score between 0 and 1, and we want to find a clustering that minimizes the sum of the similarity scores of pages in different clusters plus the sum of (one minus the similarity score) for pages in the same cluster. Another example arises in meta-search engines for Web search, where we want to get robust rankings that are not sensitive to the various shortcomings and biases of individual search engines by combining the rankings of the individual search engines [4].

* Supported by NSF grant CCF-0514628.
** Supported by NSF grant CCF-0514628.

More formally, in the *weighted minimum feedback arc set problem in tournaments*, we are given a set of elements V, nonnegative weights $w_{(i,j)}$ and $w_{(j,i)}$ for each pair of distinct elements i and j, and we want to find a permutation π that minimizes the weight of pairs of elements out of order with respect to the permutation, i.e. $\sum_{\pi(i)<\pi(j)} w_{(j,i)}$. We say the weights satisfy *probability constraints* if for any pair i, j, $w_{(i,j)} + w_{(j,i)} = 1$, or the *triangle inequality* if for any triplet i, j, k, $w_{(i,j)} + w_{(j,k)} \geq w_{(i,k)}$. We will sometimes refer to this problem as the *ranking* problem. In the *rank aggregation problem*, the input is a collection of orderings of V, and $w_{(i,j)}$ is the fraction of orderings in which i is ordered before j; note that these weights obey both the probability constraints and triangle inequality. In the *constrained* version of the ranking problem, we are also given a partial order P as input and the output permutation π must be consistent with P, i.e. if $(i, j) \in P$ then $\pi(i) < \pi(j)$.

In the *weighted clustering problem*, we are given a set of elements V, and values $w^+_{\{i,j\}}$ and $w^-_{\{i,j\}}$ for every distinct pair of elements i, j. We want to find a clustering minimizing $\sum_{i,j \text{ in different clusters}} w^+_{\{i,j\}} + \sum_{i,j \text{ in same cluster}} w^-_{\{i,j\}}$. We say the weights satisfy probability constraints if for every $i, j \in V$, $w^+_{\{i,j\}} + w^-_{\{i,j\}} = 1$. We will say the weights satisfy the triangle inequality if for every triple i, j, k, $w^-_{\{i,j\}} + w^-_{\{j,k\}} \leq w^-_{\{i,k\}}$ and $w^+_{\{i,j\}} + w^-_{\{j,k\}} \leq w^+_{\{i,k\}}$. The problem where exactly one of $w^+_{\{i,j\}}$ and $w^-_{\{i,j\}}$ is 1 (and the other 0) is called *correlation clustering*. The clustering problem corresponding to rank aggregation, in which we want to aggregate a collection of clusterings of the same set of elements, is called *consensus clustering*. We can also have a constrained version of the weighted clustering problem by giving as input sets of pairs of items P^+ and P^-, where pairs in P^+ must be in the same output cluster, while pairs in P^- must be in different output clusters.

Both rank aggregation and consensus clustering are NP-hard [4, 5], so the more general problems of ranking or clustering with weights that satisfy the triangle inequality or probability constraints, or both, are also NP-hard.

Ailon, Charikar and Newman [1] give the first constant-factor approximation algorithms for the unconstrained ranking and clustering problems with weights that satisfy either triangle inequality constraints, probability constraints, or both. Their algorithms are randomized and based on Quicksort: the algorithms recursively generate a solution by choosing a random vertex as "pivot" and ordering all other vertices with respect to the pivot vertex according to some criterion. For example, in the first type of algorithm they give for the ranking problem, a vertex j is ordered before the pivot k if $w_{(j,k)} \geq w_{(k,j)}$ or ordered after k otherwise. Next, the algorithm recurses on the two instances induced by the vertices before and after the pivot.

In the case of rank aggregation and consensus clustering, a folklore result is that returning the best of the input rankings or clusterings is a 2-approximation algorithm. Ailon, Charikar and Newman also show that one can obtain better approximation factors for rank aggregation and consensus clustering by returning the best of their algorithm's solution and the best input ranking/clustering.

For instance, for rank aggregation, they obtain a randomized $\frac{11}{7}$-approximation algorithm using their first type of algorithm, and a randomized $\frac{4}{3}$-approximation algorithm using their second, LP-based, algorithm.

There has been a good deal of follow-up work since the Ailon et al. paper. Ailon and Charikar [6] extend the pivot-based approximation algorithms for clustering to hierarchical clustering. Coppersmith, Fleischer, and Rudra [7] give a simple greedy 5-approximation algorithm for the ranking problem when weights obey the probability constraints. Van Zuylen, Hegde, Jain and Williamson [2] give deterministic variants of the pivoting algorithms in Ailon et al. and Ailon and Charikar and extend them to the constrained versions of these problems. All but one of their algorithms require solving an LP relaxation of the problem, and their techniques do not extend to the improved results in the Ailon et al. paper for rank aggregation and consensus clustering. They do give a combinatorial approximation algorithm for the ranking problem when weights obey the probability constraints, with a performance guarantee of 4 in the unconstrained case, and 6 for constrained problems.

Kenyon-Mathieu and Schudy [8] show that there exists a polynomial-time approximation scheme for unconstrained weighted feedback arc set in tournaments with weights satisfying $b \leq w_{(i,j)} + w_{(j,i)} \leq 1$ for all $i, j \in V$ for some $b > 0$. Note that this includes problems satisfying the probability constraints and hence includes the rank aggregation problem as a special case. Their approximation scheme assumes the availability of a solution with cost that is not more than a constant factor α from optimal. To get a $(1 + \epsilon)$-approximate solution, the running time of their algorithm is doubly exponential in $\frac{1}{\epsilon}$, $\frac{1}{b}$ and α.

Ailon [3] considers the *partial rank aggregation* problem, which was introduced by Fagin, Kumar, Mahdian, Sivakumar and Vee [9, 10]. Unlike full rank aggregation, the input rankings do not have to be permutations of the same set of elements. Instead, input rankings are allowed to be top-m rankings, i.e. permutations of only a subset of the elements (in which case we make the natural assumption that the unranked elements all share the position after the last ranked element), or the rankings may be p-ratings, i.e. mappings from V to $\{1, \ldots, p\}$, as is the case for example in movie rankings. More precisely, a partial ranking of a set of elements V is a function $\pi : V \to \{1, \ldots, |V|\}$. If π is bijective, it is a *full ranking*. We will say the distance between two partial rankings π_1 and π_2 is the number of pairs i, j such that $\pi_1(i) < \pi_1(j)$, and $\pi_2(i) > \pi_2(j)$. The goal of partial rank aggregation is, given ℓ partial rankings of V, to output a permutation of the elements of V that minimizes the sum of the distances from the ℓ input rankings. Note that the output is required to be a permutation, and cannot be a partial ranking. Ailon [3] generalizes and improves some of the results from Ailon et al. to partial rank aggregation. He shows that perturbing the solution to the linear programming relaxation and using these perturbed values as probabilities gives a randomized $\frac{3}{2}$-approximation algorithm for partial rank aggregation. Since his analysis only uses the fact that the weights satisfy the triangle inequality, this also yields $\frac{3}{2}$-approximation algorithm for ranking with triangle inequality constraints on the weights.

1.1 Our Results

Our goals in obtaining the results for this paper were twofold. First, we wanted to do an implementation study of the various pivoting algorithms. But none of the deterministic pivoting algorithms thus far are especially practical. The PTAS of Kenyon-Mathieu and Schudy is of theoretical interest only. Although the deterministic algorithms of Van Zuylen et al. are polynomial-time, with the exception of their combinatorial algorithm for ranking with probability constraints, they require solving a linear program with $O(n^2)$ variables and $O(n^3)$ constraints for $n = |V|$, which with standard LP packages is likely to be slow for even moderate values of n. Thus we give purely combinatorial, deterministic pivoting algorithms. For weights obeying the triangle inequality, we give a 2-approximation algorithm, whose analysis is particularly simple. In the case of rank aggregation and consensus clustering, we give an $\frac{8}{5}$-approximation algorithm. The $\frac{8}{5}$-approximation algorithm extends to the partial rank aggregation problem as well. This gives the first combinatorial algorithm for partial rank aggregation with an approximation guarantee less than 2. The running time of our combinatorial algorithms is $O(n^3)$ compared to $O(n^2)$ for their randomized counterparts in Ailon et al.

Second, we wished to give deterministic algorithms matching the best randomized algorithms of Ailon et al. and Ailon in the case of rank aggregation and partial aggregation, and correlation and consensus clustering. It is a fundamental question whether everything computable in randomized polynomial time is computable in deterministic polynomial time (something that the recent PRIMES in P result by Agrawal, Kayal, and Saxena [11] provided some additional evidence for). The techniques from Ailon et al. and Ailon are not amenable to standard techniques of derandomization, but we show (in the current paper and [2]) that we can amortize in place of the expectation and make the randomized algorithm deterministic. In particular, we show how to derandomize the $\frac{4}{3}$-approximation algorithm of Ailon et al. for rank aggregation and consensus clustering, the $\frac{5}{2}$-approximation algorithm of Ailon et al. for ranking and clustering with probability constraints, and the $\frac{3}{2}$-approximation algorithm of Ailon for partial rank aggregation. These algorithms invoke an interesting two-step derandomization, in which we first choose a pivot so as to minimize a ratio of expectations; then we apply the method of conditional expectations to decide how to order (cluster) the elements with respect to the pivot.

Finally, we show that if the weights satisfy the triangle inequality, then any approximation result that holds for unconstrained ranking or clustering problems also holds for constrained problems, by showing a sequence of local moves that remove any violations of the constraints and do not increase the cost of the solution.

In the remainder of the paper, we will only discuss our results for ranking, and not for clustering. It is straightforward to translate these results to the clustering setting (see Ailon et al. [1] and Van Zuylen et al. [2]).

Ranking				
	prob. constr.	triangle ineq.	full rank agg.	partial rank agg
Combin.	4(ZHJW), *3(ACN)*	2(ZW), *2(ACN)*,	$\frac{8}{5}$(ZW), $\frac{11}{7}$*(ACN)*	$\frac{8}{5}$(ZW)
LP based	$\frac{5}{2}$(ZW), $\frac{5}{2}$*(ACN)*	$\frac{3}{2}$(ZW), $\frac{3}{2}$*(A)*	$\frac{4}{3}$(ZW), $\frac{4}{3}$*(ACN)*	$\frac{3}{2}$(ZW), $\frac{3}{2}$*(A)*
PTAS	$1+\varepsilon$(KS)		$1+\varepsilon$(KS)	

Clustering			
	prob. constr.	triangle ineq.	consensus clustering
Combinatorial	6(ZHJW), *3(ACN)*	2(ZW), *2(ACN)*	$\frac{8}{5}$(ZW), $\frac{11}{7}$*(ACN)*
LP based	$\frac{5}{2}$(ZW), $\frac{5}{2}$*(ACN)*	2(ZHJW), *2(ACN)*	$\frac{4}{3}$(ZW), $\frac{4}{3}$*(ACN)*

Table 1. The table summarizes the best known approximation guarantees. Italicized entries are expected approximation guarantees from randomized algorithms. 'A' refers to Ailon [3], 'ACN' refers to Ailon, Charikar and Newman [1], 'KS' refers to Kenyon-Mathieu and Schudy [8], 'ZHJW' refers to Van Zuylen, Hegde, Jain, and Williamson [2] and 'ZW' refers to this paper.

2 Combinatorial Pivoting Algorithms

Given an instance of the weighted feedback arc set problem in tournaments, suppose we form a tournament $G = (V, A)$ by including arc (i, j) only if $w_{(i,j)} \geq w_{(j,i)}$ (breaking ties arbitrarily). This is called the majority tournament in Ailon et al. [1]. Clearly, if the tournament is acyclic, then it corresponds to an optimal permutation: the cost for pair i, j in any solution is at least $\min\{w_{(i,j)}, w_{(j,i)}\}$, and this lower bound is met for every pair.

Ailon, Charikar and Newman [1] propose a simple algorithm to obtain a permutation that costs at most 3 times the optimum if the weights satisfy the triangle inequality, or at most 2 times the optimum if the weights satisfy both triangle inequality and probability constraints. Their algorithm, FAS-Pivot, is given below. We use the following notation: We denote by $G(V')$ the subgraph of G induced by $V' \subset V$. If π_1 and π_2 are permutations of disjoint sets V_1, V_2, we let π_1, π_2 denote the concatenation of the two permutations.

FAS-Pivot$(G = (V, A))$

Pick a pivot $k \in V$.
$V_L = \{i \in V : (i, k) \in A\}, V_R = \{i \in V : (k, i) \in A\}$.
Return FAS-Pivot$(G(V_L))$, k, FAS-Pivot$(G(V_R))$.

In Ailon, Charikar and Newman's algorithm, a pivot is chosen randomly. In our deterministic versions of this algorithm, we will propose different ways of choosing the pivot, depending on the information we have about the input.

For a pair i, j with (i, j) in the majority tournament A, we will let $w_{ij} = w_{(i,j)}$ and $\bar{w}_{ij} = w_{(j,i)}$. Note that this implies that $w_{ij} = \max(w_{(i,j)}, w_{(j,i)})$ and $\bar{w}_{ij} = \min(w_{(i,j)}, w_{(j,i)})$. Then if a pair i, j is ordered according to A, the cost

incurred by the algorithm is $\bar{w}_{ij}$, and for each pair not ordered according to A, the cost is w_{ij}. We will call the first type of arcs "forward arcs" and the second "backward arcs".

Ailon, Charikar and Newman bound the expected cost for the backward arcs if the pivot is chosen randomly. Subsequently, Van Zuylen, Hegde, Jain and Williamson showed that one can obtain a deterministic version of this algorithm by first solving a linear programming relaxation, and then carefully choosing the pivot vertex based on the solution to the linear program. We show that if the weights satisfy the triangle inequality, then we can give a deterministic algorithm that does not require us to solve a linear program and that achieves the same guarantees as in Van Zuylen et al. for this case. The idea of our algorithm is to use $\bar{w}_{ij}$ as a "budget" for vertex pair i, j, and to show that we can always choose a pivot so that the cost of the backward arcs created by pivoting on this vertex is at most twice the budget for these arcs.

Theorem 1. *There exists a deterministic combinatorial 2-approximation algorithm for weighted feedback arc set in tournaments with triangle inequality.*

Proof. We use the algorithm described above, but specify how to choose a good pivot. For a given pivot k, we let $T_k(V) \subset A$ be the set of arcs that become backward by pivoting on k when the set of vertices in the recursive call is V. Our choice of pivot is then:

$$\text{Pick } k \in V \text{ minimizing}^1 \frac{\sum_{(i,j)\in T_k(V)} w_{ij}}{\sum_{(i,j)\in T_k(V)} \bar{w}_{ij}} \tag{1}$$

As was observed in [2], (i, j) is a backward arc if (k, i) and (j, k) in A, in other words, exactly when (i, j) is in a directed triangle in A with the pivot k. Therefore $T_k(V)$ contains exactly the arcs that are in a directed triangle with k in $G(V)$. The cost incurred for the arcs in $T_k(V)$ if k is the pivot is equal to $\sum_{(i,j)\in T_k(V)} w_{ij}$, and we have a lower bound on the cost in any feasible solution for these vertex pairs of $\sum_{(i,j)\in T_k(V)} \bar{w}_{ij}$.

Let T be the set of directed triangles in $G(V)$, and for a triangle $t = (i, j), (j, k), (k, i)$, let $w(t) = w_{ij} + w_{jk} + w_{ki}$ and let $\bar{w}(t) = \bar{w}_{ij} + \bar{w}_{jk} + \bar{w}_{ki}$. If we sum $\sum_{(i,j)\in T_k(V)} w_{ij}$ over all $k \in V$, i.e. $\sum_{k\in V} \sum_{(i,j)\in T_k(V)} w_{ij}$, then we count w_{ij} exactly once for every pivot k such that $(i, j), (j, k), (k, i)$ is a directed triangle, hence $\sum_{k\in V} \sum_{(i,j)\in T_k(V)} w_{ij} = \sum_{t\in T} w(t)$. Similarly, $\sum_{(i,j)\in T_k(V)} \bar{w}_{ij} = \sum_{t\in T} \bar{w}(t)$.

Now, note that for $t = (i, j), (j, k), (k, i)$, by the triangle inequality on the weights, $w_{ij} = w_{(i,j)} \leq w_{(i,k)} + w_{(k,j)} = \bar{w}_{ki} + \bar{w}_{jk}$, or more generally $w_a \leq \sum_{a'\in t: a'\neq a} \bar{w}_{a'}$ for any $a \in t$. Hence $w(t) \leq 2\bar{w}(t)$. Thus we have that

$$\sum_{k\in V} \sum_{(i,j)\in T_k(V)} w_{ij} = \sum_{t\in T} w(t) \leq 2 \sum_{t\in T} \bar{w}(t) = \sum_{k\in V} \sum_{(i,j)\in T_k(V)} \bar{w}_{ij}.$$

[1] Throughout this work, we define a ratio to be 0 if both numerator and denominator are 0. If only the denominator is 0, we define it to be ∞.

Hence, there exists some k such that $\sum_{(i,j)\in T_k(V)} w_{ij} \leq 2\sum_{(i,j)\in T_k(V)} \bar{w}_{ij}$, and the cost incurred for the backward arcs when pivoting on k is not more than 2 times the lower bound on the cost for these vertex pairs. □

As in Ailon, Charikar and Newman [1], we can do better in the case of rank aggregation. In fact, we will extend the ideas from Ailon, Charikar and Newman [1], and Ailon [3] to give a combinatorial $\frac{8}{5}$-approximation algorithm for partial rank aggregation.

In the partial rank aggregation problem, we are given ℓ partial rankings $\pi_1, \ldots, \pi_\ell$ where $\pi_k : V \to \{1, \ldots, |V|\}$ for $k = 1, \ldots, \ell$. In the (full) rank aggregation problem, $\pi_1, \ldots, \pi_\ell$ are bijective. We let $w_{(i,j)} = \frac{1}{\ell}\sum_{k=1}^{\ell} \mathbf{1}_{(\pi_k(i)<\pi_k(j))}$ and note that the weights for the partial rank aggregation problem satisfy the triangle inequality.

A well-known 2-approximation for full rank aggregation outputs one of the input permutations at random: the expected cost for pair i, j is $2w_{(i,j)}w_{(j,i)}$ which is not more than $2\bar{w}_{ij}$. It follows that returning the best input permutation is also a 2-approximation algorithm.

Ailon [3] proposes the algorithm RepeatChoice for partial rank aggregation. Let $\pi_1, \ldots, \pi_\ell$ be the input rankings; π will be our final output ranking. We start by setting $\pi(i) = 1$ for all $i \in V$. Then we repeatedly choose an input ranking π_k uniformly at random without replacement; we check each $i, j \in V$ and if $\pi(i) = \pi(j)$ but $\pi_k(i) < \pi_k(j)$, we modify π so that now $\pi'(i) < \pi'(j)$. We can do this by setting $\pi'(h) = \pi(h)$ if $\pi(h) \leq \pi(i)$ and $\pi'(h) = \pi(h) + 1$ if $h = j$ or $\pi(h) > \pi(i)$.

Note that π may not yet be a full ranking: We will say that $i \equiv j$ if $\pi_k(i) = \pi_k(j)$ for every input ranking π_k. At the end of the RepeatChoice procedure, we arbitrary break the ties between i, j, $i \equiv j$, so that π is a full ranking. For ease of exposition, we will henceforth assume that there are no pairs i, j such that $\pi_k(i) = \pi_k(j)$ for all $k = 1, \ldots, \ell$, although our results also hold if such pairs do exist.

The probability that i is ranked before j is $\frac{w_{(i,j)}}{w_{(i,j)}+w_{(j,i)}}$ which incurs a cost of $w_{(j,i)}$. Since i is either ranked before j, or j before i, the expected cost for pair i, j is $\frac{2w_{(j,i)}w_{(i,j)}}{w_{(i,j)}+w_{(j,i)}}$. If we define the majority tournament $G = (V, A)$ as above, and let $w_{ij} = \max\{w_{(i,j)}, w_{(j,i)}\}$ and $\bar{w}_{ij} = \max\{w_{(i,j)}, w_{(j,i)}\}$ as before, then the total expected cost for the permutation returned by RepeatChoice is $\sum_{(i,j)\in A} \frac{2w_{ij}\bar{w}_{ij}}{w_{ij}+\bar{w}_{ij}} \leq 2\sum_{(i,j)\in A} \bar{w}_{ij}$. Ailon shows that this algorithm can be derandomized.

Ailon, Charikar and Newman show that the best of their algorithm's solution and the best input permutation is a $\frac{11}{7}$-approximation algorithm for (full) rank aggregation. We show that a similar guarantee can be given for our deterministic algorithm, and moreover that this result also holds for partial rank aggregation, i.e. the best of our algorithm's solution, and the solution given by RepeatChoice gives a combinatorial $\frac{8}{5}$-approximation algorithm for partial rank aggregation.

Theorem 2. *There exists a deterministic combinatorial $\frac{8}{5}$-approximation algorithm for partial rank aggregation.*

Proof. We again use the algorithm described above, but specify a different way of choosing a good pivot. Let $T_k(V) \subset A$ be the set of arcs that become backward by pivoting on k when the set of vertices in the recursive call is V. Let $\alpha_{ij} = w_{(i,j)} + w_{(j,i)} = w_{ij} + \bar{w}_{ij}$, and note that $\bar{w}_{ij} = \alpha_{ij} - w_{ij}$. In our partial rank aggregation algorithm, we use the following rule to choose a pivot vertex.

$$\text{Pick } k \in V \text{ minimizing } \frac{\sum_{(i,j)\in T_k(V)} \left(\frac{8}{5} w_{ij} - \frac{6}{5}\frac{w_{ij}^2}{\alpha_{ij}}\right)}{\sum_{(i,j)\in T_k(V)} (\alpha_{ij} - w_{ij})} \tag{2}$$

We charge our pivoting algorithm $\frac{2}{5}$ times the cost of the solution it generates, plus $\frac{3}{5}$ times the expected cost of the permutation returned by RepeatChoice. We will show that the total cost charged is not more than $\frac{8}{5}$ times the lower bound given by $\sum_{(i,j)\in A} \bar{w}_{ij} = \sum_{(i,j)\in A} (\alpha_{ij} - w_{ij})$. Taking the better solution from the pivoting solution and the (derandomized) RepeatChoice solution then gives a $\frac{8}{5}$-approximation algorithm.

The expected cost incurred by pair i, j in RepeatChoice is $2\frac{w_{(i,j)}w_{(j,i)}}{w_{(i,j)}+w_{(j,i)}} = 2\frac{w_{ij}(\alpha_{ij}-w_{ij})}{\alpha_{ij}}$. The cost if i, j is ranked according to the majority tournament is $\bar{w}_{ij} = \alpha_{ij} - w_{ij}$, and if it is not ordered according to the majority tournament, the cost is w_{ij}. Hence the charge for a forward arc is

$$\frac{2}{5}(\alpha_{ij} - w_{ij}) + \frac{6}{5}\frac{w_{ij}(\alpha_{ij} - w_{ij})}{\alpha_{ij}} \le \frac{2}{5}(\alpha_{ij} - w_{ij}) + \frac{6}{5}(\alpha_{ij} - w_{ij}) = \frac{8}{5}(\alpha_{ij} - w_{ij})$$

The charge for a backward arc is

$$\frac{2}{5} w_{ij} + \frac{6}{5}\frac{w_{ij}(\alpha_{ij} - w_{ij})}{\alpha_{ij}} = \frac{8}{5} w_{ij} - \frac{6}{5}\frac{w_{ij}^2}{\alpha_{ij}}.$$

We will show that there always exists a pivot k such that the ratio in (2) is at most $\frac{8}{5}$. This implies that the combined charge for the arcs that become backward in one iteration can be bounded by $\frac{8}{5}$ times the lower bound on their combined cost in any feasible solution. Since the charge for a forward arc between a vertex pair is also at most $\frac{8}{5}$ times the lower bound for the vertex pair, the total charge at the end of the algorithm is at most $\frac{8}{5}\sum_{(i,j)\in A}(\alpha_{ij} - w_{ij}) = \sum_{(i,j)\in A} \bar{w}_{ij}$, which is at most $\frac{8}{5}$ times the optimal cost.

To show that a pivot with ratio at most $\frac{8}{5}$ exists, we use the same techniques as before. Let T again be the set of directed triangles in A, and for a triangle $t = (i,j),(j,k),(k,i)$, let $w(t) = w_{ij} + w_{jk} + w_{ki}, \alpha(t) = \alpha_{ij} + \alpha_{jk} + \alpha_{ki}$ and $z(t) = \frac{w_{ij}^2}{\alpha_{ij}} + \frac{w_{jk}^2}{\alpha_{jk}} + \frac{w_{ki}^2}{\alpha_{ki}}$. Note that

$$\sum_{k\in V} \sum_{(i,j)\in T_k(V)} \left(\frac{8}{5} w_{ij} - \frac{6}{5}\frac{w_{ij}^2}{\alpha_{ij}}\right) = \frac{8}{5}\sum_{t\in T} w(t) - \frac{6}{5}\sum_{t\in T} z(t),$$

and

$$\sum_{k\in V} \sum_{(i,j)\in T_k(V)} (\alpha_{ij} - w_{ij}) = \sum_{t\in T} \alpha(t) - \sum_{t\in T} w(t).$$

Note that by the triangle inequality constraints, $w(t) = w_{(i,j)} + w_{(j,k)} + w_{(k,i)} \leq w_{(i,j)} + w_{(j,k)} + (w_{(k,j)} + w_{(j,i)}) = \alpha_{ij} + \alpha_{jk}$. Similarly, we get that $w(t) \leq \alpha_{ij} + \alpha_{ki}$ and $w(t) \leq \alpha_{jk} + \alpha_{ki}$. Adding these constraints, we get that $w(t) \leq \frac{2}{3}\alpha(t)$.

By these observations and Claim 3 below, we can conclude that $\frac{8}{5}\sum_{t\in T} w(t) - \frac{6}{5}\sum_{t\in T} z(t) \leq \frac{8}{5}\left(\sum_{t\in T}\alpha(t) - \sum_{t\in T} w(t)\right)$, and hence there exists some $k \in V$ such that the ratio in (2) is at most $\frac{8}{5}$. □

Claim 3. For $w = (w_1, w_2, w_3)$, and $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ such that $0 \leq w_i \leq \alpha_i \leq 1$ for $i = 1, 2, 3$, and $\sum_{i=1}^{3} w_i \leq \frac{2}{3}\sum_{i=1}^{3}\alpha_i$:

$$16\sum_{i=1}^{3} w_i - 6\sum_{i=1}^{3}\frac{w_i^2}{\alpha_i} - 8\sum_{i=1}^{3}\alpha_i \leq 0$$

Proof. The proof uses standard techniques, and for space reasons is deferred to the full version.

Lemma 4. *The algorithms in Theorem 1 and 2 can be implemented in $O(n^3)$ time.*

Proof. We maintain a list of the directed triangles in G for which all three vertices are currently contained in a single recursive call, and for each vertex we maintain the total cost for the vertex pairs that get a backward arc if pivoting on that vertex and the total budget for these pairs (i.e. the numerator and denominator of (1) resp. (2)). If we disregard the time needed to obtain and update this information, then a single recursive call takes $O(n)$ time, and there are at most $O(n)$ iterations, giving a total of $O(n^2)$. Initializing the list of triangles and the numerator and denominator of (1) or (2) for each vertex takes $O(n^3)$ time. Over all recursive calls combined, the time needed to update the list of directed triangles, and the numerator and denominator of (1) or (2) is $O(n^3)$: After each pivot, we need to remove all triangles that either contain the pivot vertex, or contain (i, j) where i and j are separated into different recursive calls, and for each triangle removed from the list, we need to update the numerator and denominator of (1) or (2) for the three vertices in the triangle. Assuming the list of triangles is linked to the vertices and arcs contained in it and vice versa, finding a triangle that contains a certain vertex or arc, removing it, and updating the numerator and denominator for the vertices contained in it, can be done in constant time. Finally, note that each triangle is removed from the list exactly once. □

3 Two-Step Derandomization of LP-based Pivoting Algorithms

We now show how to extend the ideas from [2] to derandomize the randomized rounding algorithm in Ailon et al. [1], and the perturbed version in Ailon [3]. In particular, this allows us to obtain a deterministic $\frac{5}{2}$-approximation algorithm

for ranking with probability constraints, and a $\frac{3}{2}$-approximation algorithm for partial rank aggregation. Combined with the ideas from Theorem 2, this also allows us to obtain a deterministic $\frac{4}{3}$-approximation algorithm for full rank aggregation.

The linear program we will use is the following:

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i<j} \Big(x_{(i,j)} w_{(j,i)} + x_{(j,i)} w_{(i,j)} \Big) & \\
\text{s.t.} & x_{(i,j)} + x_{(j,k)} + x_{(k,i)} \geq 1 & \text{for all distinct } i,j,k \\
(LP) & x_{(i,j)} + x_{(j,i)} = 1 & \text{for all } i \neq j \\
& x_{(i,j)} \geq 0 & \text{for all } i \neq j
\end{array}
$$

Given an optimal solution x to this LP, we will write $c_{ij} = c_{ji} = x_{(i,j)} w_{(j,i)} + x_{(j,i)} w_{(i,j)}$.

Given an optimal solution x to the linear programming relaxation, in Ailon et al. [1] a vertex i is ordered to the left of the pivot k with probability $x_{(i,k)}$, and to the right of the pivot with probability $x_{(k,i)} = 1 - x_{(i,k)}$. In Ailon [3], the probabilities are perturbed by a function h that satisfies $h(1-x) = 1-h(x)$. Since we can always take h to be the identity, we assume without loss of generality that the probabilities are always given by $h(x)$.

FASLP-Pivot(V, x)

Pick a (random) pivot $k \in V$.
Set $V_L = \emptyset, V_R = \emptyset$.
For all $i \in V, i \neq k$,
    with probability $h(x_{(i,k)})$: add i to V_L,
    else (with probability $h(x_{(k,i)})$): add i to V_R.
Return FASLP-Pivot(V_L, x), k, FASLP-Pivot(V_R, x).

We will say an arc (i,j) is a forward arc if the vertices i,j were in the same recursive call in which one of them was the pivot, and we will say an arc (i,j) is backward if the vertices i,j were in the same recursive call, in which some vertex $k \neq i,j$ was the pivot, and i was added to V_L and j was added to V_R. Note the difference from our previous definition of forward and backward arcs.

Let $T_k(V)$ be the set of arcs that become backward in a recursive call on V when k is the pivot. Note that $T_k(V)$ is a random set, since V_L, V_R are random sets. In particular, $(i,j) \in T_k(V)$ with probability $h(x_{(i,k)})h(x_{(k,j)})$, and the expected cost for the arcs that become backward arcs is $\mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} w_{(j,i)}\Big] = \sum_{i\in V\setminus\{k\}} \sum_{j\in V\setminus\{k\}} h(x_{(i,k)})h(x_{(k,j)})w_{(j,i)}$.

We derandomize the algorithm in two steps. First we choose a pivot k such that ratio of the expected cost for the arcs in $T_k(V)$ and $\mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} c_{ij}\Big]$ is as small as possible. Then we use the method of conditional expectation [12] to assign the vertices in $V\setminus\{k\}$ to V_L or V_R.

We define the following notation: Let V_L, V_R, V' be a partition of $V\backslash\{k\}$, and let $\mathbb{E}\Big[B_k(V)|V_L, V_R\Big]$ be the expected total cost incurred in an iteration of FASLP-Pivot for the backward and forward arcs when pivoting on k conditioned on the vertices in V_L and V_R being ordered to the left and right of k respectively (and the vertices in V' are ordered left or right with probability $h(x_{(i,k)})$ and $h(x_{(k,i)})$). Let $\mathbb{E}\Big[C_k(V)|V_L, V_R\Big]$ be the expected total LP contribution for the vertex pairs that are in forward or backward arcs in an iteration of FASLP-Pivot when pivoting on k, again conditioned on V_L, V_R. Note that the conditional expected cost for backward arcs is

$$\mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} w_{(j,i)}|V_L, V_R\Big] = \sum_{i\in V_L}\sum_{j\in V_R} w_{(j,i)} + \sum_{i\in V'}\sum_{j\in V'} h(x_{(i,k)})h(x_{(k,j)})w_{(j,i)} + \sum_{i\in V_L}\sum_{j\in V'} h(x_{(k,j)})w_{(j,i)} + \sum_{i\in V'}\sum_{j\in V_R} h(x_{(i,k)})w_{(j,i)},$$

and the conditional expected LP budget for backward arcs can be computed similarly. Hence we can easily compute these conditional expectations and we get

$$\mathbb{E}\Big[B_k(V)|V_L, V_R\Big] = \mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} w_{(j,i)}|V_L, V_R\Big] + \sum_{i\in V_L} w_{(k,i)} + \sum_{i\in V_R} w_{(i,k)} + \sum_{i\in V'}\Big(h(x_{(i,k)})w_{(k,i)} + h(x_{(k,i)})w_{(i,k)}\Big),$$

$$\text{and } \mathbb{E}\Big[C_k(V)|V_L, V_R\Big] = \mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} c_{ij} \mid V_L, V_R\Big] + \sum_{i\in V\backslash\{k\}} c_{ik}.$$

DerandFASLP-Pivot(V, x)

Pick $k \in V$ minimizing $\dfrac{\mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} w_{(j,i)}\Big]}{\mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} c_{ij}\Big]}$.
Set $V_L = \emptyset, V_R = \emptyset$.
For $i \in V\backslash\{k\}$
  If $\dfrac{\mathbb{E}\Big[B_k(V)\big|\ V_L \cup \{i\}, V_R\Big]}{\mathbb{E}\Big[C_k(V)\ \big|\ V_L \cup \{i\}, V_R\Big]} \le \dfrac{\mathbb{E}\Big[B_k(V)\ \big|\ V_L, V_R \cup \{i\}\Big]}{\mathbb{E}\Big[C_k(V)\ \big|\ V_L, V_R \cup \{i\}\Big]}$
    add i to V_L,
  else
    add i to V_R.
Return DerandFASLP-Pivot(V_L, x), k, DerandFASLP-Pivot(V_R, x).

Lemma 5. *If $h(x_{(i,j)})w_{(j,i)} + h(x_{(j,i)})w_{(i,j)} \le \alpha c_{ij}$ and there always exists a pivot k with ratio at most α, then DerandFASLP-Pivot is an α-approximation algorithm.*

Proof. In our analysis of DerandFASLP-Pivot, it will be convenient to consider an "intermediate" derandomization of FASLP-Pivot. Let DFASLP-Pivot be the algorithm that chooses a pivot as in DerandFASLP-Pivot, but then randomly assigns vertices to V_L and V_R as in FASLP-Pivot.

We think of $c_{ij} = x_{(i,j)}w_{(j,i)} + x_{(j,i)}w_{(i,j)}$ as the "LP budget" for vertex pair i, j. We say a pair i, j gets decided in an iteration of DFASLP-Pivot if it either gets a forward arc (i.e. one of i, j is the pivot) or a backward arc (i.e. one of them gets assigned to V_L and one to V_R). Under the assumptions in the lemma, the expected cost for the pairs that get decided in an iteration of DFASLP-Pivot is at most α times the expected LP budget for these pairs: the total expected cost for the pairs that get decided in an iteration of DFASLP-Pivot with pivot k is $\mathbb{E}\Big[B_k(V) \mid V_L = \emptyset, V_R = \emptyset\Big] =$

$$\sum_{i \in V\setminus\{k\}} \big(h(x_{(i,k)})w_{(k,i)} + h(x_{(k,i)})w_{(i,k)}\big) + \mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} w_{(j,i)}\Big],$$

and by the assumptions of the lemma this expected cost is at most

$$\alpha\Big(\sum_{i \in V\setminus\{k\}} c_{ik} + \mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} c_{ij}\Big]\Big) = \alpha\mathbb{E}\Big[C_k(V) \mid V_L = \emptyset, V_R = \emptyset\Big].$$

Hence DFASLP-Pivot is an expected α-approximation algorithm. By standard conditional expectation arguments, we know that if we consider some vertex $i \in V\setminus(V_L \cup V_R \cup \{k\})$ and $\mathbb{E}\Big[B_k(V) \mid V_L, V_R\Big] \leq \alpha\mathbb{E}\Big[C_k(V) \mid V_L, V_R\Big]$, then we can add i to either V_L or V_R and maintain the invariant that $\mathbb{E}\Big[B_k(V) \mid V_L, V_R\Big] \leq \alpha\mathbb{E}\Big[C_k(V) \mid V_L, V_R\Big]$. Therefore, DerandFASLP-Pivot returns a a partition V_L, V_R of $V\setminus\{k\}$ such that the cost of ordering the vertices in V_L and V_R to the left and right of k respectively, is at most α times the total LP budget of the pairs that get decided in that iteration. □

Note that one can show that there always exists a pivot with ratio at most α, by showing that $\sum_{k\in V} \mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} w_{(j,i)}\Big] \leq \alpha \sum_{k\in V} \mathbb{E}\Big[\sum_{(i,j)\in T_k(V)} c_{ij}\Big]$ for any feasible LP solution. Using similar observations as in the proof of Theorem 1, it is possible to reduce this inequality to a certain inequality on triples of vertices. Using Lemma 13 in [1] this gives us Corollary 6 and using an inequality from [3] this implies corollory 7.

Corollary 6. *DerandFASLP-Pivot with $h(x) = x$ is a $\frac{5}{2}$-approximation algorithm for weighted feedback arc set in tournaments with probability constraints.*

Corollary 7. *DerandFASLP-Pivot is a $\frac{3}{2}$-approximation algorithm for partial rank aggregation and for ranking with weights that obey triangle inequality if we use $h(x) = \begin{cases} \frac{3}{4}x, & 0 \leq x \leq \frac{1}{3} \\ \frac{3}{2}x - \frac{1}{4}, & \frac{1}{3} < x \leq \frac{2}{3} \\ \frac{3}{4}x + \frac{1}{4}, & \frac{2}{3} < x \leq 1 \end{cases}$ as proposed in [3].*

We can obtain a $\frac{4}{3}$-approximation algorithm for rank aggregation by using the techniques from Theorem 2 to show that the best of DerandFASLP-Pivot and picking a random input permutation is within $\frac{4}{3}$ of optimal. Similar to the technique in the proof of Theorem 2 we replace the weight $w_{(i,j)}$ by $\frac{2}{3}w_{(i,j)} + \frac{1}{3}(2w_{(i,j)}w_{(j,i)})$, and show that DerandFASLP-Pivot returns a solution for which the cost with respect to these new weights is at most $\frac{4}{3}$ times optimal. Since the cost with respect to these new weights is a convex combination of the cost of the solution with respect to the original weights, and the cost of a randomly chosen input permutation, we get that the best of the algorithm's solution and the best input permutation is a $\frac{4}{3}$-approximation algorithm. For space reasons the details of the proof are deferred to the full version.

Theorem 8. *There exists a deterministic $\frac{4}{3}$-approximation algorithm for rank aggregation.*

4 Constrained problems

We now consider ranking problems where we are also given a partial order P, and the output permutation π must be consistent with P; in other words, if $(i,j) \in P$ then $\pi(i) < \pi(j)$. We make the natural assumption that the weights are consistent with P, i.e. if $(i,j) \in P$ then $w_{(j,i)} = 0$. It is possible to use similar techniques as in [2] to ensure that the algorithms in Section 2 return a feasible solution. However, a stronger result is given by the following lemma, which says that if the weights satisfy the triangle inequality, then any permutation that is not consistent with P is not a local minimum. We thank Frans Schalekamp for suggesting that this may be the case. This means that all results in this paper, except for the result in Corollary 6, also hold for constrained problems.

Lemma 9. *Given weights that satisfy the triangle inequality,a permutation π, and a partial order P such that $w_{(j,i)} = 0$ for $(i,j) \in P$, then we can find a permutation π' that is consistent with P and costs not more than π.*

Proof. Let $(i,j) \in P$ and suppose $\pi(j) < \pi(i)$. We call such (i,j) violated. Let $K(i,j)$ be the set of vertices k such that $\pi(j) < \pi(k) < \pi(i)$, and let (i^*, j^*) be a violated pair such that for any vertex $k \in K(i^*, j^*)$ it is the case that $(j^*, k) \notin P$ and $(k, i^*) \notin P$. (Note that by transitivity of P, if a violated pair exists, then there exists a violated pair that satisfies this condition.)

Consider the permutation π' we obtain by moving j^* to the position just after i^* with probability $p = \frac{1}{2}$ or otherwise moving i^* to the position just before j^*. Note that (i^*, j^*) is not violated in π' and no new violations are created.

The expected difference in the cost of permuations π' and π is given by

$$w_{(j^*,i^*)} - w_{(i^*,j^*)} + \frac{1}{2} \sum_{k \in K(i^*,j^*)} (w_{(j^*,k)} - w_{(k,j^*)} + w_{(k,i^*)} - w_{(i^*,k)})$$

$$\leq w_{(j^*,i^*)} - w_{(i^*,j^*)} + \frac{1}{2} \sum_{k \in K(i^*,j^*)} (2w_{(j^*,i^*)}) = -w_{(i^*,j^*)} \leq 0,$$

where the first inequality follows from the triangle inequality, and the last equality follows since $w_{(j^*,i^*)} = 0$. Hence either moving j^* to the position just after i^* or moving i^* to the position just before j^* does not increase the cost of the permutation, and has fewer violations. □

References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. In: STOC 2005. 684–693
2. van Zuylen, A., Hegde, R., Jain, K., Williamson, D.P.: Deterministic pivoting algorithms for constrained ranking and clustering problems. In: SODA 2007. 405–414
3. Ailon, N.: Aggregation of partial rankings, p-ratings and top-m lists. In: SODA 2007. 415–424
4. Dwork, C., Kumar, S.R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: WWW 2001. 613–622
5. Wakabayashi, Y.: The complexity of computing medians of relations. Resenhas **3**(3) (1998) 323–349
6. Ailon, N., Charikar, M.: Fitting tree metrics: Hierarchical clustering and phylogeny. In: FOCS 2005. 73–82
7. Coppersmith, D., Fleischer, L., Rudra, A.: Ordering by weighted number of wins gives a good ranking for weighted tournaments. In: SODA 2006. 776–782
8. Kenyon-Mathieu, C., Schudy, W.: How to rank with few errors: A PTAS for weighted feedback arc set on tournaments. In: STOC 2007. 95–103
9. Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., Vee, E.: Comparing partial rankings. SIAM J. Discret. Math. **20**(3) (2006) 628–648
10. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. SIAM J. Discret. Math. **17**(1) (2003) 134–160
11. Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P. Ann. of Math. (2) **160**(2) (2004) 781–793
12. Alon, N., Spencer, J.: The Probabilistic Method. Wiley Interscience (1992)