How to Rank with Few Errors

A PTAS for Weighted Feedback Arc Set on Tournaments*

Claire Kenyon-Mathieu Warren Schudy†

Brown University
Computer Science
Providence, RI

ABSTRACT

We present a polynomial time approximation scheme (PTAS) for the minimum feedback arc set problem on tournaments. A simple weighted generalization gives a PTAS for Kemeny-Young rank aggregation.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms

Algorithms, Theory

Keywords

Approximation algorithm, Feedback arc set, Kemeny-Young rank aggregation, Max acyclic subgraph, Polynomial-time approximation scheme, Tournament graphs

1. INTRODUCTION

Suppose you ran a chess tournament, everybody played everybody, and you wanted to use the results to rank everybody. Unless you were really lucky, the results would not be acyclic, so you could not just sort the players by who beat whom. A natural objective is to find a ranking that minimizes the number of upsets, where an upset is a pair of players where the player ranked lower on the ranking beats the player ranked higher. This is the minimum feedback arc set (FAS) problem on tournaments. Our main result is a polynomial time approximation scheme (PTAS) for this problem. A simple weighted generalization gives a PTAS for Kemeny-Young rank aggregation.

*An extended version is available [31]

†Corresponding author. Email: ws@cs.brown.edu

STOC'07, June 11–13, 2007, San Diego, California, USA.

For general directed graphs, the FAS problem consists of removing the fewest number of edges so as to make the graph acyclic, and has applications such as scheduling [21] and graph layout [42, 13] (see also [34, 5, 27]). This problem has been much studied, both in the mathematical programming community [43, 23, 24, 27, 32] and the approximation algorithms community [33, 38, 19]. The best known approximation ratio is $O(\log n \log\log n)$. Unfortunately the problem is NP-hard to approximate better than 1.36... [28, 15].

A *tournament* is the special case of directed graphs where every pair of vertices is connected by exactly one of the two possible directed edges. For tournaments, the FAS problem also has a long history, starting in the early 1960s in combinatorics [37, 44] and statistics [39]. In combinatorics and discrete probability, much early work [18, 35, 36, 26, 40, 41, 20] focused on worst-case tournaments, starting with Erdös and Moon and culminating with work by Fernandez de la Vega. In statistics and psychology, one motivation is *ranking by paired comparisons*: here, you wish to sort some set by some objective but you do not have access to the objective, only a way to compare a pair and see which is greater; for example, determining people's preferences for types of food. Early interesting heuristics due to Slater and Alway can be found in [39]. Unfortunately, even on tournaments, the FAS problem is still NP-hard [1, 3] (see also [11]). The best approximation algorithms achieve constant factor approximations [1, 12, 45]: 2.5 in the randomized setting [1], 3 in the deterministic setting [45, 46, 1]. Our main result is a polynomial time approximation scheme (PTAS) for this problem: thus the problem really is provably easier to approximate in tournaments than on general graphs.

Here is a weighted generalization.

Problem 1 (Weighted FAS Tournament). *Input: complete directed graph with vertex set V, $n \equiv |V|$ and nonnegative edge weights w_{ij} with $w_{ij} + w_{ji} \in [b, 1]$ for some fixed constant $b \in (0, 1]$. Output: A total order $R(x, y)$ over V minimizing $\sum_{\{x,y\} \subset V} w_{xy} R(y, x)$.*

(The unweighted problem is the special case where $w_{ij} = 1$ if (i, j) is an edge and $w_{ij} = 0$ otherwise.)

In [1], the variant where $b = 1$ is called weighted FAS tournament with probability constraints. Indeed, sampling a population naturally leads to defining w_{ij} as the probability that type i is preferred to type j. We note that all known approximation algorithms [1, 12] extend to weighted tournaments for $b = 1$.

An important application of weighted FAS tournaments is *rank aggregation*. Frequently, one has access to several

rankings of objects of some sort, such as search engine outputs [16, 17], and desires to aggregate the input rankings into a single output ranking that is similar to all of the input rankings: it should have minimum average distance from the input rankings, for some notion of distance. This ancient problem was already studied in the context of voting by Borda [6] and Condorcet [10] in the 18th century, and has aroused renewed interest in learning [9]. A natural notion of distance is the number of pairs of vertices that are in different orders: this defines the *Kemeny-Young rank aggregation* problem [29, 30]. This problem is NP-hard, even with only 4 voters [17]. Constant factor approximation algorithms are known: choosing a random (or best) input ranking as the output gives a 2-approximation; finding the optimal aggregate ranking using the footrule distance as the metric instead of the Kendall-Tau distance also gives a 2-approximation [16]. There is also a randomized 4/3 approximation algorithm [1, 2]. We improve on these results by designing a polynomial-time approximation scheme.

THEOREM 1 (PTAS). *There are randomized algorithms for minimum Feedback Arc Set on weighted tournaments and for Kemeny-Young rank aggregation that, given $\lambda > 0$, output orderings with expected cost less than $(1+\lambda)OPT$. Let $\epsilon = \lambda b$. The running time is:*

$$O\left((1/\epsilon)n^6 + n^2 2^{\tilde{O}(1/\epsilon)} + n 2^{2^{\tilde{O}(1/\epsilon)}}\right).$$

The algorithms can be derandomized at the cost of increasing the running time to $n^{2^{\tilde{O}(1/\epsilon)}}$.

Our algorithm combines several existing algorithmic tools. It uses (a) existing constant factor approximation algorithms. In addition, it uses (b) existing polynomial-time approximation schemes for the complementary maximization problem (a.k.a. max acyclic subgraph tournaments), due to Arora, Frieze and Kaplan [4] and to Frieze and Kannan [22]. Moreover, our central algorithm, Algorithm 2, also uses (c) the divide-and-conquer paradigm. Of course, this has been previously used in this setting, in partitioning algorithms in general graphs [33] and in a Quicksort-type algorithm [1], but both of these constructions are top-down, whereas our algorithm is bottom-up, which gives it a very different flavor. Finally, it also relies on (d) a simple but useful technique, used already in 1961 by Slater and Alway [39]: iteratively improve the current ordering by taking a vertex out of the ordering and moving it back at a different position. (Such single vertex moves and variants have been studied at some length since that time [25, 44, 16, 17].)

It is fortunate that the FAS problem on tournaments has an approximation scheme. The closely related problem of correlation clustering on complete graphs[1] is APX-hard [8]. The related problem of feedback *vertex* set is also hard to approximate even on tournaments, though it does have a 2.5-approximation algorithm [7].

We note that Amit Agarwal has informed us that he recently obtained similar results.

[1]This problem is identical to FAS except it deals with symmetric transitive relations instead of antisymmetric ones.

2. ALGORITHM

In this section, we present our deterministic and randomized algorithms and reduce one to the other.

Our algorithms use the sampling-based additive approximation algorithms of [4, 22] as a subroutine. For simplicity, here we use the derandomized additive error algorithm. Section 4.1 discusses how to extend our analysis to use the randomized version, yielding the runtime in Theorem 1.

THEOREM 2 (SMALL ADDITIVE ERROR). *[4, 22] There is a randomized polynomial-time approximation scheme for maximum acyclic subgraph on weighted tournaments. Given $\beta, \eta > 0$, the algorithm outputs, in polynomial time, [22]*

$$O\left(\left[n/\beta^4 + 2^{O(1/\beta^2)}\right]\log(1/\eta)\right)$$

an ordering whose cost is, with probability at least $1-\eta$, less than $OPT+\beta n^2$. The algorithm can be derandomized into an algorithm which we denote by AddApprox, which increases the runtime to $n^{O(1/\beta^4)}$.

We also use a simple type of local search. A total order R can be specified by an ordering $\pi : V \to \{1, \ldots, n\}$ such that $\pi(x)$ is the position of vertex x: $\pi(x) < \pi(y)$ iff $R(x, y)$. A *single vertex move*, given a ordering π, a vertex x and a position i, consists of taking x out of π and putting it back in position i.

All of our algorithms call the additive approximation algorithm with parameter

$$\beta = \beta(\epsilon) \equiv 9^{-\frac{1}{\epsilon}\log_{3/2}(1/\epsilon^2)}\epsilon^3 = 2^{-O(1/\epsilon \log(1/\epsilon))}.$$

Our deterministic PTAS is given in Algorithm 1. Recall from Problem 1 that b is the lower bound on $w_{xy} + w_{yx}$ for every pair $\{x, y\}$. For ease of thinking, the reader may think of b as being equal to 1. Also recall that we are seeking a $1 + \lambda = 1 + \epsilon/b$ approximation.

Our (somewhat faster) randomized PTAS is given in Algorithm 2. Curiously, if the constant factor approximation algorithm used is the sorting by indegree algorithms from [6, 12], the initial order π^{local} computed in Algorithm 2 is precisely the order returned by the heuristic algorithm created by Slater and Alway [39] in 1961!

For the analysis, it is enough to study Algorithm 2, since the result for Algorithm 1 follows as a simple corollary:

Algorithm 1 Deterministic polynomial time approximation scheme of Theorem 1 (PTAS)

Given: Fixed parameters $\epsilon > 0$ and $b \in (0, 1]$.
Input: A weighted tournament.
Round weights to integer multiples of ϵ/n^2.
$\pi \leftarrow$ constant factor approximation [1, 12].
While some move decreases cost, do it. Types of moves:

1. **Single vertex moves.** Choose vertex x and rank j, take x out of the ordering and put it back in so that its rank is j.
2. **Additive approximation.** Choose integers $i < j$; let $U = \{$vertices with rank in $[i, j]\}$. $\pi'_U \leftarrow$AddApprox(U) with parameter $\beta(\epsilon)$. Replace the restriction π_U of π to U by π'_U.

Output: π.

Algorithm 2 Randomized polynomial time approximation scheme (RPTAS) of Theorem 1. Vertical lines in left margin denote differences from Algorithm 3.

Given: Fixed parameters $\epsilon > 0$ and $b \in (0, 1]$.
Input: A weighted tournament
  Round weights to integer multiples of ϵ/n^2.
  $\pi \leftarrow$ constant factor approximation [1, 12].
  Apply single vertex moves to get a local optimum π^{local}.
Output: $\mathbf{Rec}(1, n)$

$\mathbf{Rec}(i, j) =$
  If i=j **then Return** i
  For any ℓ, m, let $V_{\ell,m} = \{$ vertices x: $\pi^{local}(x) \in [\ell, m]\}$
  $K \leftarrow \{k : |V_{ik}|, |V_{k+1,j}| \geq |V_{ij}|/3\}$.
  Choose k uniformly at random from K.
  $\rho_1 \leftarrow$ AddApprox(V_{ij}) with parameter $\beta(\epsilon)$
  $\rho_2 \leftarrow$ Concatenate $\mathbf{Rec}(i, k)$ and $\mathbf{Rec}(k + 1, j)$
  Return ρ_1 or ρ_2, whichever has lower cost.

LEMMA 3. *If Algorithm 2 outputs an ordering whose cost is $\leq (1 + \lambda)OPT$ with positive probability, then Algorithm 1 outputs an ordering with cost $\leq (1 + \lambda)OPT$.*

PROOF. Let π denote the output of Algorithm 1. Since we started with a constant factor approximation and only performed cost-improving moves, π is still a constant factor approximation, of course. Execute Algorithm 2 starting with this ordering π. Since there are no cost-decreasing single-vertex moves, $\pi^{local} = \pi$. Since Algorithm AddApprox cannot improve any subinterval U of π, the recursive process will always choose ρ_2 rather than ρ_1: in the end, we obtain $\pi^{out} = \pi$ for every execution of our Algorithm 2. Therefore it must be that π has cost $\leq (1 + \lambda)$OPT. □

3. ANALYSIS: CORRECTNESS

3.1 Rounding analysis

The following lemma shows that rounding the weights, which is done to ensure polynomial-time termination of local search, is irrelevant for the correctness analysis.

LEMMA 4 (ROUNDING). *Let C denote the cost function with the original weights, $\tilde{C}$ denote the cost function with the rounded weights, and $\tilde{OPT} = \min_\rho \tilde{C}(\rho)$ denote the value of the optimal solution to the problem with rounded weights. If $\tilde{C}(\pi) \leq (1 + \lambda)\tilde{OPT}$, then $C(\pi) \leq (1 + 3\lambda)OPT$.*

PROOF SKETCH. The proof is split into two cases, depending on whether OPT $< b(1/2 - \lambda)$. □

Thanks to Lemma 4, without loss of generality we can assume that the input weights are integer multiples of ϵ/n^2; we will make this assumption for the remainder of the paper.

3.2 Reduction to a virtual algorithm

Algorithm 2 is quite difficult to analyze directly, because it uses bottom-up recursion, and so we will instead analyze a different, top-down process, Algorithm 3. This is a virtual algorithm whose sole purpose is to guide the analysis: indeed, it takes as input parameter an optimal ordering π^*, so it cannot be a real algorithm!

For any subset S of vertices, the *Spearman's footrule distance* between the two orderings used in the specification of

Algorithm 3 Virtual algorithm. Vertical lines in left margin denote differences from Algorithm 2.

Given: Fixed parameters $\epsilon > 0$ and $b \in (0, 1]$.
Input: Weighted tournament; optimal ordering π^*
  Round weights to integer multiples of ϵ/n^2.
  $\pi \leftarrow$ constant factor approximation [1, 12].
  Apply single vertex moves to get a local optimum π^{local}.
  Let $\alpha = 9^{-r}$, with r uniform in $[0, \log_9(\epsilon^3/\beta)]$.
Output: $\pi^{out} = \mathbf{Rec}$(1,n)

$\mathbf{Rec}$(i, j) =
  If i=j **then Return** i
  For any ℓ, m, let $V_{\ell,m} = \{$vertices x : $\pi^{local}(x) \in [\ell, m]\}$
  $K \leftarrow \{k : |V_{ik}|, |V_{k+1,j}| \geq |V_{ij}|/3\}$.
  Choose k uniformly at random from K.
  $\rho_1 \leftarrow$ AddApprox(V_{ij}) with parameter β
  $\rho_2 \leftarrow$ Concatenate $\mathbf{Rec}(i, k)$ and $\mathbf{Rec}(k + 1, j)$
  If $F_{V_{i,j}} \geq \alpha\epsilon^2|V_{i,j}|^2$
    Return ρ_1 (V_{ij} is called a leaf)
  else
    Return ρ_2 (V_{ij} is called an internal node)

the stopping condition used in Algorithm 3 is defined by

$$F_S = \sum_{x \in S} |\pi^{local}(x) - \pi^*(x)|.$$

In tournament graphs, the footrule distance is related to the cost of the two orderings:

LEMMA 5. $F_V \leq (2/b)(C_V(\pi^*) + C_V(\pi^{local}))$.

PROOF. By [14]'s Theorem 2 relating Spearman's footrule and Kendall-Tau, we have:

$$\begin{aligned}
&\sum_j |\pi^{local}(j) - \pi^*(j)| \\
&\leq 2\sum_{i,j} \mathbb{1}\left(\pi^*(i) > \pi^*(j)\right) \mathbb{1}\left(\pi^{local}(i) < \pi^{local}(j)\right) \\
&\leq \frac{2}{b}\sum_{i,j}\left[\mathbb{1}\left(\pi^*(i) > \pi^*(j)\right) w_{ij} + \mathbb{1}\left(\pi^{local}(i) < \pi^{local}(j)\right) w_{ji}\right] \\
&= (2/b)(C(\pi^*) + C(\pi^{local})).
\end{aligned}$$

where

$$\mathbb{1}(p) \equiv \begin{cases} 1 & \text{if } p \text{ is true} \\ 0 & otherwise \end{cases}.$$

The second inequality follows from $w_{ij} + w_{ji} \geq b$. □

The following Lemma shows that it is sufficient to analyze Algorithm 3.

LEMMA 6. *If ${\pi^{out}}_2$ denotes the output of the Algorithm 2 and π^{out} denotes the output of the Algorithm 3, then:*

$$\forall x, \quad \Pr(cost({\pi^{out}}_2) \leq x) \geq \Pr(cost(\pi^{out}) \leq x).$$

PROOF. Couple the executions of the two algorithms so that they start with the same π, do the same sequence of single-vertex moves leading to the same π^{local}, and make the same choice of k in their recursive divide-and-conquer process when they have the same value of (i, j). The two divide-and-conquer processes have the same tree of recursive calls. By bottom-up induction on the tree nodes, the

ordering returned by the process used in Algorithm 2 has cost less than or equal to that of the ordering returned by the process used in Algorithm 3. □

We now focus on the output π^{out} of Algorithm 3, whose analysis hinges on the following main technical result, proven in Section 3.5. For any subset of vertices S, let

$$C_S(\pi) = \sum_{\{x,y\} \subseteq S, \pi(x) > \pi(y)} w_{xy}.$$

The objective is to minimize the overall cost $C_V(\pi)$, which we also denote as $C(\pi)$ (no subscript) for shorthand.

THEOREM 7. *For the orderings defined in Algorithm 3:*

$$\mathbf{E}\left[C(\pi^{out})\right] - C(\pi^*) \leq O(\epsilon/b)(C(\pi^{local}) + C(\pi^*)).$$

Since π^{local} is an improvement over the constant factor approximation π, we still have $C(\pi^{local}) \leq C(\pi) = O(C(\pi^*))$. Thus $\mathbf{E}\left[C(\pi^{out})\right] \leq C(\pi^*)(1+O(\epsilon/b))$, hence Algorithm 3 is an approximation scheme. By Lemma 6, so is Algorithm 2, and by Lemma 3, so is Algorithm 1.

The running time analysis is deferred to Section 4.2, so all we need to do is analyze the virtual algorithm.

3.3 Intuition

We are now ready for some intuition for Algorithm 3. When the optimal cost $C(\pi^*)$ is large, the additive approximation algorithm is satisfactory. When $C(\pi^*)$ is small, $C(\pi^{local})$ is also small (because it is a constant factor approximation). This implies that the footrule distance will also be small (Lemma 5), the stopping condition will not hold, and Algorithm 3 will recurse. It will perform some partition $V = V_{1,k} \cup V_{k+1,n}$, thereby irrevocably committing all vertices of $V_{1,k}$ to precede all vertices of $V_{k+1,n}$ in the output ordering. We need to show that the mistakes made in such dividing steps are not too bad.

However, since the footrule distance is small in that case, by definition of footrule, the positions of the vertices in π^{local} are typically close to their positions in π^*. When k is chosen at random, the vertices with rank $\{1, \ldots, k\}$ in π^{local} will be more or less the same as the vertices with rank $\{1, \ldots, k\}$ in π^*. Thus, we expect that only a few vertices will be misplaced during the divide step.

How costly can it be, if just one or two vertices are misplaced during the divide step? Unfortunately, it can be quite costly. For example, if a misplaced vertex x, which has rank $\ell > k$ in π^{local}, has rank 1 in π^*, then misplacing just a single vertex x may conceivably incur a cost of $\Theta(n)$. This is where the single-vertex moves come to the rescue. By optimality, π^* cannot improve its cost by moving x to position ℓ, and by local optimality, π^{local} cannot improve its cost by moving x to position 1. It must be that about half of the edges between x and $V_{1,k}$ are directed towards x and about half are directed away from x, and so, it does not really matter whether we place x in position 1 or in position ℓ. This explains why the algorithm makes sense intuitively.

3.4 Reduction to analyzing π^{local}

We first reduce the problem to analyzing π^{local} instead of π^{out}. Since the two orderings only differ inside the leaves of the recursion tree, we have:

LEMMA 8. *For every sequence of random choices of the algorithm, we have*

$$C_V(\pi^{out}) - C_V(\pi^{local}) = \sum_{S \text{ leaf}} (C_S(\pi^{out}) - C_S(\pi^{local})).$$

To prove Theorem 7, we replace $C_V(\pi^{out}) - C_V(\pi^*)$ by:

$$(C_V(\pi^{out}) - C_V(\pi^{local})) + (C_V(\pi^{local}) - C_V(\pi^*)).$$

We then apply Lemma 8 and replace, in each leaf S, the quantity $C_S(\pi^{out}) - C_S(\pi^{local})$ by

$$(C_S(\pi^{out}) - C_S(\pi^*)) - (C_S(\pi^{local}) - C_S(\pi^*)).$$

Now all we need to do is analyze, on the one hand, the expectation of

$$C_V(\pi^{local}) - C_V(\pi^*) - \sum_{S \text{ leaf}} (C_S(\pi^{local}) - C_S(\pi^*)), \quad (1)$$

(this only involves π^{local}, not π^{out}, and will be the bulk of the proof), and on the other hand,

$$\sum_{S \text{ leaf}} (C_S(\pi^{out}) - C_S(\pi^*)), \quad (2)$$

which we now deal with.

The following lemma motivates the stopping condition of Algorithm 3.

LEMMA 9 (LEAF NODES). *Let S be a leaf of the recursion tree. Then $C_S(\pi^{out}) - C_S(\pi^*) \leq \epsilon F_S$.*

PROOF. If $|S| = 1$ then the claim is trivial. Otherwise, using Theorem 2, the definition of tree leaves, and defining $\alpha_{min} = \beta/\epsilon^3 \leq \alpha$:

$$\begin{aligned} C_S(\pi^{out}) - C_S(\pi^*) &\leq C_S(\pi^{out}) - \mathrm{OPT}_S \\ &\leq \beta|S|^2 = \alpha_{min}\epsilon^3|S|^2 \\ &\leq (\alpha_{min}\epsilon^3)(F_S/(\alpha\epsilon^2)) \leq \epsilon F_S. \end{aligned}$$

□

Using Lemma 9 in Expression (2), along with the obvious fact $\sum_S F_S = F_V$, and applying Lemma 5 yields

$$\sum_{S \text{ leaf}} (C_S(\pi^{out}) - C_S(\pi^*)) \leq (2\epsilon/b)(C_V(\pi^*) + C_V(\pi^{local})),$$

which is one of the contributions to the right-hand side of Theorem 7.

In order to complete the proof of Theorem 7, all that is left to do is to argue that in expectation, Expression (1) is also $O(\epsilon/b)(C_V(\pi^*) + C_V(\pi^{local}))$.

3.5 Cost decomposition

To analyze Expression (1), we need to compare π^{local} to π^*. Define the *displacement graph* with vertex set $[1, n]$ (corresponding to the ranks), and arc set $(\pi^{local}(x), \pi^*(x))$ for every x. For example, here is the displacement graph if π^* is in almost perfect agreement with π^{local}, except for switching the first and last vertices in the ordering:

The local optimality of π^{local} with respect to single-vertex moves helps bound the difference in cost between π^* and π^{local}.

Let $f(e)$ denotes the cost, for edge $e = \{x, y\}$ of modifying π^{local} by switching the relative order of x and y:

$$f(x,y) = \begin{cases} w_{xy} - w_{yx} & \pi^{local}(x) < \pi^{local}(y) \\ w_{yx} - w_{xy} & o.w. \end{cases}$$

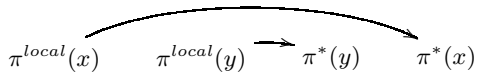
Let T_S denote the (negation of the) sum of the costs of the single vertex moves that modify π^{local} by taking $x \in S$ and moving it from position $\pi^{local}(x)$ to position $\pi^*(x)$. More precisely:

$$T_S = T_S^{(1)} + T_S^{(2)}, \text{ with}$$

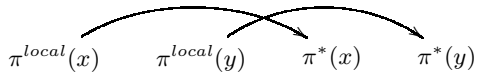
$$\begin{aligned} T_S^{(1)} &= \sum_{x \in S} \sum_{y \in S : \pi^{local}(x) < \pi^{local}(y) \leq \pi^*(x)} -f(x,y) \\ T_S^{(2)} &= \sum_{x \in S} \sum_{y \in S : \pi^*(x) \leq \pi^{local}(y) \leq \pi^{local}(x)} -f(x,y) \end{aligned}$$

The two terms $T_S^{(1)}$ and $T_S^{(2)}$ correspond to arcs from left to right and right to left respectively in the displacement graph. (The negation is there because we are trying to upperbound $C(\pi^{local}) - C(\pi^*)$, not $C(\pi^*) - C(\pi^{local})$.) T_S considers the displacement arcs one at a time: this is in a sense a first order approximation to the change in cost, since it ignores the interaction between several displacements.

How does T_S compare to $C_S(\pi^{local}) - C_S(\pi^*)$? Some pairs $\{x, y\}$ contribute the same to T_S as to $C_S(\pi^{local}) - C_S(\pi^*)$, such as:



For some other pairs, there is a discrepancy, such as the following pair which appears in the sum for T_S but not in the sum for $C_S(\pi^{local}) - C_S(\pi^*)$:



This is what we call a "crossing pair" (see Section 3.7), and we need a corrective term, which we will denote by Φ_V, taking those into account. Thus we define:

$$\Phi_S = C_S(\pi^{local}) - C_S(\pi^*) - T_S.$$

The following two Lemmas, proved in later subsections, analyze the increase of the first order (T) and second order (Φ) effects from the leaves to the root.

Recall that $\alpha \in [\alpha_{min}, 1]$ with $\alpha_{min} = 9^{-\log_{3/2}(1/\epsilon^2)/\epsilon}$. The choice of α and of the random k's determines the random choices in the execution. Let B denote the choices of k for an execution of Algorithm 3 with $\alpha = 1$. Smaller α leads to earlier termination, so B also defines the random choices made for any choice of α. Thus, for a given π^{local}, the random choices are determined by (B, α).

LEMMA 10. *For a random execution defined by a random choice of (B, α), we have*

$$T_V - \mathbf{E}_{B,\alpha}\left[\sum_{S \text{ leaf}} T_S\right] \leq 14\epsilon F_V.$$

LEMMA 11. *Let $\gamma = \frac{\sqrt{5}/3}{1-\sqrt{5}/3}$. For any α, for a random execution defined by a random choice of (B), we have*

$$\Phi_V - \mathbf{E}_B\left[\sum_{S \text{ leaf}} \Phi_S\right] \leq 96\gamma\epsilon F_V.$$

Now we can bound Expression (1). Since Lemma 11 is true for every α, it is also true in expectation over α. Adding Lemma 10 and substituting the definition of Φ to both sides yields:

$$\begin{aligned} C_V(\pi^{local}) - C_V(\pi^*) - \mathbf{E}\left[\sum_{S \text{ leaf}} (C_S(\pi^{local}) - C_S(\pi^*))\right] \\ \leq (14 + 96\gamma)\epsilon F_V = O(\epsilon) F_V; \end{aligned}$$

applying Lemma 5 concludes the proof of Theorem 7. It only remains to prove Lemmas 11 and 10.

3.6 Proof of Lemma 10 (analyzing T_V)

In this part, we prove Lemma 10. We split $T_V - \sum_{S \text{ leaf}} T_S$ into separate terms for each vertex y, and bound each term by the *displacement* $|\pi^*(y) - \pi^{local}(y)|$ of that vertex times ϵ.

Given two vertices x and y, let $E(x)$ be the event that $\pi^{local}(x)$ is between $\pi^{local}(y)$ and $\pi^*(y)$ but x is not in the leaf containing y. First we see that by definition of T_V and $\{T_S\}$,

$$T_V - \mathbf{E}_{B,\alpha}\left[\sum_{S \text{ leaf}} T_S\right] = \mathbf{E}_{B,\alpha}\left[\sum_y \sum_{x : E(x)} f(x,y)\right].$$

Let $s(y)$ be the *size* of the leaf containing y (number of vertices in that set). We can split the sum for each y into several parts depending on the relative size of the leaf containing y and $|\pi^*(y) - \pi^{local}(y)|$. The cases correspond to

- *small* leaves: $|\pi^*(y) - \pi^{local}(y)| > s(y)/\epsilon$, and
- *big* leaves: $\epsilon s(y) > |\pi^*(y) - \pi^{local}(y)|$,
- *intermediate* leaves: $\epsilon s(y) \leq |\pi^*(y) - \pi^{local}(y)| \leq s(y)/\epsilon$.

Small leaves are easy thanks to local optimality.

LEMMA 12 (SMALL LEAVES). *For any execution (determined by B and α), we have*

$$\sum_{y \in \text{ small leaf}} \sum_{x : E(x)} -f(x,y) \leq \epsilon F_V.$$

PROOF SKETCH. Moving y cannot improve π^{local}. □

We handle the big leaves by proving that the probability that $\pi^{local}(y)$ and $\pi^*(y)$ are in different leaves (and hence able to contribute to the sum) is $O(\epsilon)$.

LEMMA 13 (BIG LEAVES). *For any α and for a random B, we have:*

$$\mathbf{E}_B\left[\sum_{y\in \text{ big leaf}}\sum_{x:E(x)} -f(x,y)\right] \leq 12\epsilon F_V.$$

PROOF SKETCH. Let α be fixed. If $\pi^{local}(x)$ is between $\pi^{local}(y)$ and $\pi^*(y)$, but x is not in leaf(y), then the vertex whose rank in π^{local} is $\pi^*(y)$ is also not in leaf(y). Thus we can bound the expression on the left hand side by the expectation (over the random tree construction) of

$$\sum_y |\pi^*(y)-\pi^{local}(y)|\cdot \mathbb{1}\left(\begin{array}{c} y\in \text{ big leaf} \\ \text{and } \pi^{local^{-1}}(\pi^*(y))\notin \text{ leaf}(y)\end{array}\right).$$

One can that for any vertex y, the event E_1 that "$y\in$ big leaf and $\pi^{local^{-1}}(\pi^*(y))\notin$ leaf(y)" has probability at most 9ϵ over the sequence B of random choices defining the decomposition. The intuition is that the random choice of k is unlikely to land between $\pi^{local}(y)$ and $\pi^*(y)$. □

We bound the contribution of the intermediate leaves by using the variation in α to force a variation in the leaf size, making it unlikely that a given vertex y will be in an intermediate leaf.

LEMMA 14 (INTERMEDIATE LEAVES). *For any fixed B and for a random α, we have*

$$\mathbf{E}_\alpha\left[\sum_{y\in \text{ intermediate leaf}}\sum_{x:E(x)} -f(x,y)\right] \leq \epsilon F_V.$$

PROOF SKETCH. Let $B=(k_S)$ be fixed.

As in the beginning of the proof of Lemma 13, we can bound the expression on the left hand side by the expectation (over the random choice of α) of

$$\sum_y \begin{cases} 0 & \text{if } s(y)=n \\ \sum_{x:E(x)} -f(x,y) & \text{if } s(y)=1 \\ |\pi^*(y)-\pi^{local}(y)|\cdot & \\ \quad \mathbb{1}(y\in \text{ intermediate leaf and} & \\ \quad\quad \pi^{local^{-1}}(\pi^*(y))\notin \text{ leaf}(y)) & \text{otherwise.}\end{cases}$$

One can argue that for any y, the event that "$y\in$ intermediate leaf, $\pi^{local^{-1}}(\pi^*(y))\notin$ leaf(y) and $1<s(y)<n$" has low probability; but here for the first time, the probabilistic space is over the random definition of α. □

3.7 Proof of Lemma 11 (analyzing Φ_V)

First, we need an algebraic formula expressing Φ_V in terms of contributions of vertex pairs. We say that a pair of vertices x,y such that $\pi^{local}(x)<\pi^{local}(y)$ is a *crossing pair* if exactly one of the following four cases occur:

$$\begin{cases} \pi^{local}(x) & <\pi^*(y) & <\pi^*(x) & <\pi^{local}(y) \\ \pi^*(y) & \leq\pi^{local}(x) & \leq\pi^{local}(y) & \leq\pi^*(x) \\ \pi^*(x) & \leq\pi^*(y) & \leq\pi^{local}(x) & \leq\pi^{local}(y) \\ \pi^{local}(x) & \leq\pi^{local}(y) & \leq\pi^*(x) & \leq\pi^*(y)\end{cases}$$

We define δ_{xy} to be equal to 1 in the first case, -1 in the other three cases, and to be equal to 0 for non-crossing pairs. Intuitively, $\{x,y\}$ is a crossing pair if, when we move x continuously from position $\pi^{local}(x)$ to position $\pi^*(x)$, we pass exactly one of $\{\pi^{local}(y),\pi^*(y)\}$.

LEMMA 15 (MISTAKE DECOMPOSITION). *We have:*

$$\Phi_S = \sum_{x,y\in S} -f(x,y)\cdot\delta_{xy}$$

PROOF. $\Delta C\equiv C_S(\pi^{local})-C_S(\pi^*)=\sum -f(x,y)$, where the sum is over pairs x,y in S such that $\pi^{local}(x)<\pi^{local}(y)$ and $\pi^*(x)>\pi^*(y)$. To prove the Lemma, it suffices to show that each term $f(x,y)$ appears with the same coefficient in ΔC as it does in $T_S+\Phi_S$. First rewrite $T_S^{(2)}$ by swapping the names x and y and using the fact that f is symmetric to yield $T_S^{(2\prime)}=\sum -f(x,y)$, where the sum is now over pairs x,y in S such that $\pi^*(y)\leq\pi^{local}(x)<\pi^{local}(y)$. Note that every pair (x,y) in the sums ΔC, $T_S^{(1)}$, $T_S^{(2\prime)}$ and Φ_S has $\pi^{local}(x)<\pi^{local}(y)$. We divide the pairs (x,y) into seven sets based on the truth of the inequalities $\pi^*(x)>\pi^*(y)$, $\pi^*(x)\geq\pi^{local}(y)$, and $\pi^*(y)\leq\pi^{local}(x)$ (The eighth possibility never happens because it would imply a contradiction with $\pi^{local}(x)<\pi^{local}(y)$.) The seven cases are shown in the following table. T indicates that the inequality in the heading is true, integers indicate the coefficient of $f(x,y)$ in the sums, and dots indicate false or zero.

$\pi^*(x)>\pi^*(y)$	$\pi^*(x)\geq\pi^{local}(y)$	$\pi^{local}(x)\geq\pi^*(y)$	ΔC	T_1	T_2	Φ
·	·	·	·	·	·	·
·	·	T	·	·	1	-1
·	T	·	·	1	·	-1
T	·	·	1	·	·	1
T	·	T	1	·	1	·
T	T	·	1	1	·	·
T	T	T	1	1	1	-1

□

Our proof of Lemma 11 is by induction on the nodes of the recursion tree. To analyze crossing pairs, the number of vertices such that the displacement arc $(\pi^{local}(x),\pi^*(x))$ crosses k is relevant. Let

$$\psi_S^L=|\{\,x\in S|\ \pi^*(x)\leq k<\pi^{local}(x)\,\}|, \text{ and}$$

$$\psi_S^R=|\{\,x\in S|\ \pi^{local}(x)\leq k<\pi^*(x)\,\}|.$$

LEMMA 16 (CORE). *Let $S=V_{i,j}$, let L be the set of $x\in S$ such that $\pi^{local}(x)\leq k$, and $R=S\setminus L$, where $V_{i,j}$ and k are defined as in Algorithm 3. Then:*

$$\Phi_S-\mathbf{E}_k\left[\Phi_L+\Phi_R\right]\leq\frac{32}{|S|}\sum_{x\in S}|\pi^*(x)-\pi^{local}(x)|\psi_S^*.$$

where $\psi_S^*=\max_{k\in S}\max(\psi_S^L(k),\psi_S^R(k))$

PROOF. Apply Lemma 15 to Φ_S,Φ_L and Φ_R. Restrict attention to pairs x,y such that $\pi^{local}(x)<\pi^{local}(y)$ (the other pairs have $\delta_{xy}=0$.) Since $|\delta_{xy}|\leq 1$ and $|f(x,y)|\leq 1$, we can write

$$\Phi_S-\Phi_L-\Phi_R\leq|\{(x,y)\in L_k\times R_k,\ \text{crossing pair}\}|,$$

where we write $L=L_k$ and $R=R_k$ to make the dependency in k explicit. Now, it is easy to see that the set K from which Algorithm 3 randomly chooses k satisfies $|K|\geq|V_{ij}|/4$ whenever $i\neq j$. We thus have:

$$\mathbf{E}_k\left[\Phi_S-\Phi_L-\Phi_R\right]\leq$$
$$\frac{4}{|S|}|\{(x,y,k):(x,y)\in L_k\times R_k,\ \text{crossing pair}\}|.$$

In such a triple (x, y, k), we must have that k is either between $\pi^{local}(x)$ and $\pi^*(x)$ or between $\pi^{local}(y)$ and $\pi^*(y)$. Moreover, in order for $\{x, y\}$ to be a crossing pair, it must be that y cross over either $\pi^{local}(x)$ or over $\pi^*(x)$, so y is counted in $\psi_S^L(\pi^{local}(x))$ or in $\psi_S^R(\pi^{local}(x))$ or in $\psi_S^L(\pi^*(x))$ or in $\psi_S^R(\pi^*(x))$. Taking the union of these cases yields the Lemma. □

LEMMA 17. $\psi_S^* \leq F_S^{1/2}$.

PROOF SKETCH. $i \mapsto \psi_S(i)$ is a 1-Lipschitz function (it changes by at most 1 when going from i to $i+1$), and its integral is bounded by F_S. It is easy to see that these observations imply that the maximum value of this function cannot be more than $\sqrt{F_S}$. □

COROLLARY 18.

$$\Phi_S - \mathbf{E}_k\left[\Phi_L + \Phi_R\right] \leq \frac{32}{|S|} F_S^{3/2}.$$

At this point we are prepared to show that at any one level the mistakes made are bounded by 32ϵ times the optimum cost. To do this, note that by the stopping condition $\sqrt{F_S}/|S| < \epsilon$, so $F_S^{3/2}/|S| < \epsilon F_S$. This shows that any one level of the tree does not contribute excessive mistakes. The following lemma proves that the sum of mistakes over all of the levels is dominated by the nodes near the leaves of the tree.

LEMMA 19 (INDUCTIVE). *For every $\alpha \in [\alpha_{min}, 1]$,*

$$\Phi_V \leq \mathbf{E}_B\left[\sum_{S \text{ leaf}} \left[96\gamma\epsilon F_S + \Phi_S\right]\right],$$

where $\gamma = \frac{\sqrt{5}}{3-\sqrt{5}}$.

PROOF. Let $\alpha \in [\alpha_{min}, 1]$ be fixed. If V is a leaf, then the statement is true. Else, given α, we first prove by induction that for any $l \leq m$, with $U = \{\, x \mid \ell \leq \pi^{local}(x) \leq m \,\}$ and $\mathbf{E}\left[\cdot | U \in B\right]$ meaning expectation conditioned on U being a leaf or internal node in execution tree B:

$$\Phi_U \leq \mathbf{E}_B\left[\sum_{\substack{S \text{ internal} \\ \text{descendant of } U}} 32\frac{F_S^{3/2}}{|S|} + \sum_{\substack{S \text{ leaf} \\ \text{descendant of } U}} \Phi_S \middle| U \in B\right]. \quad (3)$$

The base case of U being a leaf is trivial.

Let k be the random variable used at U to decompose U into L and R. Let $X_k = \Phi_U - \Phi_L - \Phi_R$. Let B_1 be the random sequence used to construct the decomposition of the left child, and B_2 be the random sequence used to construct the decomposition of the right child, so that $B = (k, B_1, B_2)$. Note that given an interval U, $\Phi_U = \mathbf{E}_B\left[\Phi_U\right]$ is deterministic. Therefore:

$$\Phi_U \mathbf{E}_B\left[X_k + \Phi_L + \Phi_R\right] = \mathbf{E}_k\left[X_k\right] + \mathbf{E}_k\left[\Phi_L + \Phi_R\right].$$

Using the induction hypothesis for fixed L, R we have:

$$\Phi_L + \Phi_R \leq \mathbf{E}_{B_1,B_2}\left[\sum_{S \text{ internal desc. of } L \text{ or } R} 32\frac{F_S^{3/2}}{|S|} + \sum_{S \text{ leaf desc. of } L \text{ or } R} \Phi_S \middle| U, L, R \in B\right].$$

Take expectation over k and rewriting:

$$\mathbf{E}_k\left[\Phi_L + \Phi_R\right] \leq \mathbf{E}_B\left[\sum_{S \text{ internal strict desc. of } U} 32\frac{F_S^{3/2}}{|S|} + \sum_{S \text{ leaf desc. of } U} \Phi_S \middle| U \in B\right].$$

Adding $\mathbf{E}_k\left[X_k\right] \leq 32F_U^{3/2}/|U|$ (from Corollary 18) to this and combining $32F_U^{3/2}/|U|$ with the sum over internal nodes completes the induction, proving Equation 3

The rest of the proof is deterministic. We prove that for every tree decomposition B, the argument of the expectation on the right hand side of Equation 3 with $\ell, m = 1, n$ and $U = V$ is bounded by $O(1)\epsilon F_V$. Therefore the expectation must also be bounded. The proof uses the following algebraic claim which is easily checked. Recall that $\gamma = \frac{\sqrt{5}/3}{1-\sqrt{5}/3}$. If $0 \leq F_1 \leq F$, $s > 0$, and $s_1 > 0$ such that $s_1, (s - s_1) \geq s/3$, then:

$$\frac{F^{3/2}}{s} < \sqrt{5}/3\left(\frac{F_1^{3/2}}{s_1} + \frac{(F - F_1)^{3/2}}{s - s_1}\right).$$

Thus, for every internal node S, we have:

$$F_S^{3/2}/|S| \leq \sqrt{5}/3(F_{S_1}^{3/2}/|S_1| + F_{S_2}^{3/2}/|S_2|).$$

Summing over the recursion tree, we obtain:

$$\sum_{S \text{ internal}} \frac{F_S^{3/2}}{|S|} \leq \sum_{S \text{ leaf}} \frac{F_S^{3/2}}{|S|}(\sqrt{5}/3 + (\sqrt{5}/3)^2 + \cdots)$$

which is bounded by $\gamma \sum_{S \text{ leaf}} \frac{F_S^{3/2}}{|S|}$. If S is a leaf, then let P be the parent of S.[2] The parent is not a leaf, so by the stopping condition $\sqrt{F_P}/|P| \leq \alpha\epsilon \leq \epsilon$. Thus:

$$\frac{F_S^{3/2}}{|S|} = F_S\frac{\sqrt{F_S}}{|S|} \leq F_S\frac{\sqrt{F_P}}{|P|}\frac{|P|}{|S|} \leq 3F_S\frac{\sqrt{F_P}}{|P|} \leq 3\epsilon F_S.$$

Summing yields the lemma. □

Combining Lemma 19 with the simple fact that $\sum_{S \text{ leaf}} F_S = F_V$ proves Lemma 11.

4. RUNNING TIME

4.1 Improved running time: Randomization

The running time of Algorithms 2 and 1 can be improved somewhat by replacing the deterministic additive error algorithm with the randomized one. To use the randomized version of the AddApprox algorithm set $\eta = \delta/n$ for Algorithm 2 and $\eta = \delta\epsilon/n^4$ for Algorithm 1. Consider the event

[2] If S is the root, no parent is available but in that case the lemma is trivially true anyway.

E_0 stating that "During execution of the algorithm, every call to the randomized version of AddApprox yields a result within the stated bounds." Each call fails with probability at most η. As shown in Section 4.2, there are at most n (resp. n^4/ϵ) such calls, so event E_0 has probability at least $1-\delta$. Modify the analysis by assuming throughout that event E_0 holds and do the analysis conditioned on E_0.

Using the randomized version of the AddApprox algorithm introduces a complication because the proof of Lemma 3 implicitly assumes determinism. This is easily remedied by reusing the random numbers used the previous time randomized AddApprox was called on the same vertices.

4.2 Analysis: Running time

Both Algorithm 1 and 2 have a greedy local search phase. The preprocessing step ensures that the edge weights are integer multiples of ϵ/n^2, so the cost decreases by at least ϵ/n^2 at each iteration. Since the cost is always between n^2 and zero, the total number of iterations is bounded by n^4/ϵ.

The single vertex move local search phase takes time $O(n^2 \cdot n^4/\epsilon)$, where the first term comes from the time to search for an improving single vertex move and the second is a bound on the number of iterations.

The runtime is dominated by the calls to the additive error algorithm. There are at most n calls for the randomized version (Algorithm 2) and at most $n^2 \cdot n^4/\epsilon$ for the deterministic one (Algorithm 1), where the n^2 comes from the number of intervals that need to be checked each iteration to find an improving move. Thus the runtime is

$$O(n^6/\epsilon + nf(n, (1/\epsilon)^{O(1/\epsilon)}, \delta/n^4))$$

randomized and

$$O(n^6/\epsilon \cdot f(n, (1/\epsilon)^{O(1/\epsilon)}, 0))$$

deterministic, where $f(n, \beta, \eta)$ is the time required for the additive approximation algorithm to run on the problem of size n with error parameter β and failure probability η. The best-known additive approximation run time $f(n, \beta, \eta)$ is $\left(n/\beta^4 + 2^{O(1/\beta^2)}\right) \log 1/\eta$ randomized in [22], or $n^{O(1/\beta^4)}$ for deterministic ($\eta = 0$).

5. CONCLUSIONS AND FUTURE WORK

The reader may wonder whether our results can extend to the dense case, with $\Omega(n^2)$ edges but no constraint that every pair of vertices must have an edge between them. This extension is not feasible because the dense case is as hard as the general case, which is APX-hard. To see this, consider an arbitrary directed graph with n vertices that we want the feedback arc set for. Union this graph with an arbitrary acyclic dense graph on n vertices. The result is a dense graph with $2n$ vertices that is essentially equivalent to the original problem.

Interesting open questions include:

1. Does weighted feedback arc set with the triangle inequality admit a PTAS? This would give a PTAS for partial rank aggregation [2].

2. Can the runtime be improved to be only singly exponential in ϵ? The doubly exponential runtime comes from the intermediate leaves case in the T analysis (Lemma 14).

3. Is single vertex move local optimality sufficient to achieve a constant factor approximation? We have an example showing that such a constant factor is at least three. Also does sorting by indegree followed by single vertex moves have a provably better approximation factor than either one by itself?

Acknowledgements

The authors wish to thank Nir Ailon, Eli Upfal and anonymous reviewers for useful suggestions.

6. REFERENCES

[1] Nir Ailon, Moses Charikar, Alantha Newman. Aggregating inconsistent information: ranking and clustering. *ACM Symposium on Theory of Computing (STOC)*, pages 684-693, 2005.

[2] Nir Ailon. Aggregation of Partial Rankings, p-Ratings and Top-m Lists. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, 415-424.

[3] N. Alon, Ranking tournaments, *SIAM J. Discrete Math.* 20 (2006), 137-142.

[4] Sanjeev Arora, Alan Frieze, and Haim Kaplan. A New Rounding Procedure for the Assignment Problem with Applications to Dense Graph Arrangement Problems. *IEEE Symposium on Foundations of Computer Science (FOCS)* 1996: 24-33.

[5] F.B.Baker and L.J.Hubert, Applications of combinatorial programming to data analysis: seriation using asymmetric proximity measures, *British J. Math. Statis. Psych.* 30 (1977) 154-164.

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

[7] M.C. Cai, X. Deng, W. Zang. An approximation algorithm for feedback vertex sets in tournaments. *SIAM J. Computing* v. 30 No. 6 pp. 1993-2007. 2001.

[8] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *FOCS*, pages 524–533, 2003.

[9] William W. Cohen, Robert E. Schapire and Yoram Singer. Learning to order things. *Proceedings of the 1997 conference on Advances in neural information processing systems (NIPS)*: 451-457.

[10] M. Condorcet. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. 1785.

[11] Conitzer, V. Computing Slater Rankings Using Similarities Among Candidates. *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, Boston, MA, USA, 613-619. 2006.

[12] D. Coppersmith, Lisa Fleischer, Atri Rudra: Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM-SIAM Symposium on Discrete Algorithms (SODA)* 2006: 776-782.

[13] C. Demetrescu and I. Finocchi. Break the "right" cycles and get the "best" drawing. In B.E. Moret and A.V. Goldberg, editors, *Proceedings of the 2nd International Workshop on Algorithmic Engineering and Experiments (ALENEX)*, 171-182, 2000.

[14] P. Diaconis and R. Graham. Spearman's footrule as a measure of disarray. *J. of the Royal Statistical Society*, 39(2), pages 262–268, 1977.

[15] I. Dinur and S. Safra. The importance of being biased. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, pages 33-42, 2002.

[16] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation revisited. Manuscript. 2001.

[17] Cynthia Dwork, Ravi Kumar, Moni Naor, D. Sivakumar. Rank Aggregation Methods for the Web. *WWW10*. 2001.

[18] P. Erdös and J.W.Moon, On sets on consistent arcs in tournaments, *Canadian Mathematical Bulletin* 8 (1965) 269-271.

[19] G. Even, J. Naor, S. Rao, B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 62-71, 1995.

[20] Fernandez de la Vega, W. On the maximal cardinality of a consistent set of arcs in a random tournament, *J. Combinatorial Theory, Ser. B* 35:328-332 (1983).

[21] M.M.Flood. Exact and Heuristic Algorithms for the Weighted Feedback Arc Set Problem: a Special Case of the Skew-Symmetric Quadratic Assignment Problem. *Networks*, 20:1-23, 1990.

[22] Alan M. Frieze, Ravi Kannan: Quick Approximation to Matrices and Applications. *Combinatorica* 19(2): 175-220 (1999).

[23] M. Grötschel, M. Jünger, G. Reinelt. Facets of the linear ordering polytope. *Math. Programming* 33, 1985, 43-60.

[24] M. Grötschel, M. Jünger, G. Reinelt. On the acyclic subgraph polytope. *Math. Programming* 33, 1985, 28-42.

[25] R. Hassin and S. Rubinstein. Approximations for the maximum acyclic subgraph problem. *Information processing letters* 51 (1994) 133-140.

[26] H.A. Jung, On subgraphs without cycles in tournaments, *Combinatorial Theory and its Applications II*, North Holland (1970) 675-677.

[27] M. Jünger, *Polyhedral combinatorics and the Acyclic Subdigraph Problem* (Heldermann Verlag, Berlin, 1985).

[28] Karp, R., 1972. Reducibility among combinatorial problems. In: Miller, R. E., Thatcher, J. W. (Eds.), *Complexity of Computer Computations*. Plenum Press, NY, pp. 85–103.

[29] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:571-591, 1959.

[30] J. Kemeny and J. Snell. *Mathematical models in the social sciences*. Blaisdell, New York, 1962. Reprinted by MIT press, Cambridge, 1972.

[31] Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors: A PTAS for Weighted Feedback Arc Set on Tournaments ECCC Report TR06-144, Nov 2006.

[32] B. Korte, Approximative algorithms for discrete optimization problems, *Ann. Discrete Math* 4 (1979), 85-120.

[33] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 422-431. IEEE, October 1988.

[34] A. Lempel and I. Cederbaum. Minimum feedback arc and vertex set of a directed graph. *IEEE Trans. Circuit Theory*, 13(4): 399-403, 1966.

[35] K.B.Reid, On sets of arcs containing no cycles in tournaments, *Canad. Math Bulletin* 12 (1969) 261-264.

[36] K.D.Reid and E.T.Parker, Disproof of a conjecture of Erdös and Moser on tournaments, *J. Combin. Theory* 9 (1970) 225-238.

[37] S. Seshu and M.B. Reed, *Linear Graphs and Electrical Networks*, Addison-Wesley, Reading, MA, 1961. Attribute the problem of the minimum feedback arc set to Runyon.

[38] P.D.Seymour. Packing directed circuit fractionally. *Combinatorica* 15:281-288, 1995.

[39] P. Slater, Inconsistencies in a schedule of paired comparisons, *Biometrika* 48 (1961) 303-312.

[40] J. Spencer, Optimal ranking of tournaments, *Networks* 1(1971) 135-138.

[41] J. Spencer, Optimal ranking of unrankable tournaments, *Period. Math. Hungar.* 11-2 (1980) 131-144.

[42] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11(2):109-125, 1981.

[43] A.W.Tucker. On directed graphs and integer programs. Presented at 1960 *Symposium on Combinatorial Problems*, Princeton University, cited in [44].

[44] D.H. Younger. Minimum feedback arc sets for a directed graph. *IEEE Trans. Circuit Theory* 10 (1963), 238-245.

[45] A. van Zuylen. Deterministic approximation algorithms for ranking and clusterings. Cornell ORIE Tech report No. 1431. 2005.

[46] Anke van Zuylen, Rajneesh Hegde, Kamal Jain and David P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, 405-414.