How to Rank with Fewer Errors

A PTAS for Feedback Arc Set in Tournaments

CLAIRE MATHIEU and WARREN SCHUDY
Brown University

We present the first polynomial time approximation scheme (PTAS) for the minimum feedback arc set problem in tournaments. A weighted generalization gives the first PTAS for Kemeny rank aggregation. The runtime is singly exponential in $1/\epsilon$, improving on the conference version of this work, which was doubly exponential.

Categories and Subject Descriptors: G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation algorithm, Feedback arc set, Kemeny 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 in the ranking beat the player ranked higher. This is the minimum feedback arc set (FAS) problem in tournaments.

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 [Flood 1990] and graph layout [Sugiyama et al. 1981; Demetrescu and Finocchi 2000] (see also [Lempel and Cederbaum 1966; Baker and Hubert 1977; Jünger 1985]). This problem has been much studied, both in the mathematical programming community [Grötschel et al. 1985a; 1985b; Jünger 1985; Korte 1979; Newman and Vempala 2001; Newman 2004], the approximation algorithms community [Leighton and Rao 1999; Seymour 1995; Even et al. 2000; Even et al. 1998] and various applied communities [Demetrescu and Finocchi 2000; 2001; Eades et al. 1993; Dwork et al. 2001]. The best known approximation ratio is $O(\log n \log\log n)$. Unfortunately the problem is NP-hard to approximate better than 1.36 [Karp 1972; Dinur and Safra 2002]. The equivalent problem of maximizing the number of edges

Preliminary version in STOC 2007 [Kenyon-Mathieu and Schudy 2007].
Contact: ws at cs.brown.edu
Postal Address: Department of Computer Science, Brown University, 115 Waterman Street Fourth Floor, Providence RI 02912.

not removed, called *max acyclic subgraph*, has also been extensively studied [Hassin and Rubinstein 1994; Berger and Shor 1997; Charikar et al. 2007b; Newman 2001].

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 [Seshu and Reed 1961; Younger 1963] and statistics [Slater 1961]. In combinatorics and discrete probability, much early work [Erdös and Moon 1965; Reid 1969; Reid and Parker 1970; Jung 1970; Spencer 1971; 1980; Fernandez de la Vega 1983] focused on worst-case tournaments. In statistics and psychology, one motivation is *ranking by paired comparisons* [Slater 1961]: 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. Feedback arc set in tournament graphs and closely related problems have also been used in machine learning [Cohen et al. 1999; Ailon and Mohri 2008]. Unfortunately the FAS problem is NP-hard even in tournaments [Ailon et al. 2008; Alon 2006; Charbit et al. 2007] (see also [Conitzer 2006]).

Researchers have been designing algorithms for the FAS problem in tournament graphs since the dawn of computer science. Slater and Alway describe simple heuristics for the FAS tournament problem in Slater [1961] (for comparison Hoare published quicksort the same year). Coleman and Wirth [2008] compare various algorithms empirically, including an algorithm by Ailon et al. [2008] and many others with no approximation guarantees. The best previously known approximation algorithms achieve constant factor approximations: 2.5 in the randomized setting [Ailon et al. 2008] and 3 in the deterministic setting [Van Zuylen et al. 2007; Van Zuylen and Williamson 2007; Ailon et al. 2008] (see also Coppersmith et al. [2006]). 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 of feedback arc set in tournaments.

Problem 1.1 *Weighted FAS Tournament.*
Input: complete directed graph with vertex set V, $n \equiv |V|$ and non-negative edge weights w_{uv} with $b \leq w_{uv} + w_{vu} \leq 1$ for some fixed positive constant b.
Output: An ranking π minimizing $C(\pi) = \sum_{\{u,v\} \subset V : \pi(v) > \pi(u)} w_{vu}$, where a *ranking* is a bijective mapping from V to $\{1, 2, \ldots |V|\}$.

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

We now state our main theorem.

THEOREM 1.2 PTAS. *There is a randomized algorithm for minimum Feedback Arc Set on weighted tournaments. Given $\epsilon > 0$, it outputs a ranking with expected cost at most $(1 + \epsilon)OPT$. The expected running time is:*

$$O\left(n^3 \log n(\log(1/b) + 1/\epsilon)\right) + n2^{\tilde{O}(1/(\epsilon b)^6)}.$$

The algorithm can be derandomized at the cost of increasing the running time to $n^{\tilde{O}(1/(\epsilon b)^{12})}$.

We remark that our PTAS is singly exponential in $1/\epsilon$, whereas the PTAS in the conference version of this work [Kenyon-Mathieu and Schudy 2007] was doubly exponential in $1/\epsilon$.

Ailon et al. [2008] study the special-case $b = 1$, which they call *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 [Ailon et al. 2008; Coppersmith et al. 2006; Van Zuylen et al. 2007; Van Zuylen and Williamson 2007] 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 [Dwork et al. 2001], 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 [1781] and Condorcet [1785] in the 18^{th} century, and has aroused renewed interest recently [Dwork et al. 2001; Conitzer et al. 2006]. A natural notion of distance is the number of pairs of vertices that are in different orders: this defines the *Kemeny rank aggregation* problem [Kemeny 1959; Kemeny and Snell 1962]. This choice yields a maximum likelihood estimator for a certain naïve Bayes model [Young 1995]. This problem is NP-hard [Bartholdi et al. 1989], even with only 4 voters [Dwork et al. 2001]. 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 [Dwork et al. 2001; Diaconis and Graham 1977]. There is also a randomized 4/3 approximation algorithm [Ailon et al. 2008]. We improve on these results by giving a polynomial time approximation scheme (PTAS).

COROLLARY 1.3. *There is a randomized algorithm for Kemeny rank aggregation that. Given $\epsilon > 0$, it outputs a ranking with expected cost at most $(1+\epsilon)OPT$. The expected running time for n candidates is:*

$$O\left(\frac{n^3 \log n}{\epsilon}\right) + n2^{\tilde{O}(1/\epsilon^6)} + O\left(n^2 \cdot (\textit{number of voters})\right).$$

The algorithm can be derandomized at the cost of increasing the running time to $n^{\tilde{O}(1/\epsilon^{12})}$.

An important open question is whether there is a PTAS for the generalization of Kemeny rank aggregation to *partial* rankings such as search engine outputs. There is a 1.5-approximation algorithm [Ailon 2007].

It is surprising that the minimum Feedback Arc Set problem on tournaments has an approximation scheme. The related problems of correlation clustering on complete graphs[1] [Charikar et al. 2005] and of feedback *vertex* set on tournaments [Cai et al. 2001] do not have PTASs unless P=NP.

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

KwikSort(vertices S):

> Choose a vertex v uniformly at random from S
> Let L be the set of vertices u such that $w_{uv} \geq w_{vu}$ and $R = S \setminus L$.
> Return the concatenation of KwikSort(L) and KwikSort(R).

Fig. 1. KwikSort algorithm for minimum Feedback Arc Set in tournaments by Ailon et al. [2008].

Certain special cases can be solved exactly in polynomial time. Braverman and Mossel [2008] give an exact algorithm for feedback arc set when the input is generated by adding noise to a base ranking. There are also good algorithms for low-cost instances of FAS tournament and Kemeny rank aggregation (i.e. fixed-parameter tractability) [Dom et al. 2006; Betzler et al. 2008].

Other related problems include d-dimensional arrangement [Charikar et al. 2007a].

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

2. MAIN RESULTS

2.1 Algorithmic tools

Our first algorithmic tool is local search. A *single vertex move*, given an ranking π, a vertex x and a position i, consists of taking x out of π and putting it back in position i. We say a ranking is *locally optimal* if no single vertex move can improve its cost. Single vertex moves were used for FAS tournament as early as 1961 [Slater 1961]. Single vertex moves and variants were also used in [Hassin and Rubinstein 1994; Younger 1963; Dwork et al. 2001; Coleman and Wirth 2008]. Coleman and Wirth [2008] show that single vertex moves alone do not yield a constant factor approximation by giving a graph with global optimum $\Theta(n)$ and a local optimum of value $\Theta(n^2)$.

Our second algorithmic tool is the KwikSort algorithm by Ailon Charikar and Newman [2008]. Recall from Problem 1.1 that b is a lower bound on $w_{uv} + w_{vu}$ for every pair $\{u, v\}$. We show that Ailon et al. [2008]'s KwikSort algorithm, (which we reproduce for completeness in Figure 1) is a $5/b$-approximation for any $b > 0$.

Theorem 2.1. *[Ailon et al. 2008] Let w be non-negative weight function on the edges. Assume that for every $u, v, x \in V$, if $w_{uv} \geq w_{vu}$, $w_{vx} \geq w_{xv}$ and $w_{xu} \geq w_{ux}$, then $w_{uv} + w_{vx} + w_{xu} \leq \alpha \cdot \min\{w_{uv}, w_{vx}, w_{xu}\}$ for some $\alpha > 1$. Then the* KwikSort *algorithm is an α-approximation in expectation.*

Corollary 2.2. *Assume that for every pair of vertices, $b \leq w_{uv} + w_{vu} \leq 1$. Then the* KwikSort *algorithm is a $5/b$-approximation in expectation.*

Proof. (Adapted from the proof in Ailon et al. [2008] for the case $b = 1$.) Fix some triple $\{u, v, x\} \subseteq V$ with $w_{uv} \geq w_{vu}$, $w_{vx} \geq w_{xv}$ and $w_{xu} \geq w_{ux}$. We assume without loss of generality that $w_{uv} \leq w_{vx}, w_{xu}$. By weighted tournament assumption $w_{uv} + w_{vu} \geq b$ so $w_{uv} \geq b/2$, hence $2 \leq \frac{4}{b} w_{uv}$.

Therefore

$$\begin{aligned} w_{uv} + w_{vx} + w_{xu} &\leq 2 + w_{uv} \leq \left(\frac{4}{b} + 1\right) w_{uv} = \left(\frac{4}{b} + 1\right) \min\{w_{uv}, w_{vx}, w_{xu}\} \\ &\leq \frac{5}{b} \min\{w_{uv}, w_{vx}, w_{xu}\}. \end{aligned}$$

FAST-SCHEME:

Run the KWIKSORT algorithm on V to define an ranking π
Return IMPROVE(π, $\epsilon b/5$).

IMPROVE(ranking π, error tolerance η):

Set $\beta \leftarrow \frac{\eta C(\pi)}{4n \log_{3/2} n}$ and $\kappa \leftarrow \frac{\eta^2 b^3}{350 \cdot 400^2}$
Perform single vertex moves on π until none can improve the cost by more than β.
Return IMPROVEREC(V, π)

IMPROVEREC(vertices S, ranking π on S):

if $|S| = 1$ **then**
  Return π
else
  if $C(\pi) \geq \kappa |S|^2$ **then**
    Return the ranking from ADDAPPROX with $\delta = \frac{\eta}{4}\kappa$
  else
    Choose an integer k uniformly at random from $[|S|/3, 2|S|/3]$
    Let L be the set of vertices v such that $\pi(v) \leq k$ and $R = S \setminus L$
    Return concatenation of IMPROVEREC(L, π_L) and IMPROVEREC(R, π_R) where
      π_L is the ranking of L induced by π, i.e. $\pi_L(v) = \pi(v)$, and
      π_R is the ranking of R induced by π, i.e. $\pi_R(v) = \pi(v) - k$.
  end if
end if

Fig. 2. Approximation scheme for minimum Feedback Arc Set on tournaments

□

Our third and last algorithmic tool is the sampling-based approximation algorithms due to Arora, Frieze and Kaplan [2002] and to Frieze and Kannan [1999]. For completeness we use a (much simpler) algorithm based on Mathieu and Schudy [2008], which we state and analyze in Appendix A.

THEOREM 2.3. *Let $\delta > 0$ be fixed. There is a randomized algorithm* ADDAPPROX *for minimum Feedback Arc Set on weighted tournaments with expected additive error δn^2 and runtime $O\left(n^2\right) + 2^{\tilde{O}(1/\delta^2)}$.*

2.2 Algorithm

Our main algorithm FAST-SCHEME (short for Feedback Arc Set Tournament Approximation Scheme) is presented in Figure 2.

LEMMA 2.4. *Let π^{in} be the input and π^{out} be the output of the* IMPROVE *subroutine. Then:*

$$\mathbf{E}\left[C(\pi^{out})\right] \leq OPT + \eta \cdot C(\pi^{in}).$$

With this, it is easy to see that FAST-SCHEME is an approximation scheme. Indeed, by Corollary 2.2, it calls the IMPROVE subroutine for a ranking of expected cost at most $(5/b)OPT$. By definition of η and Lemma 2.4, the output ranking then has expected cost at most $(1+\epsilon)OPT$. To obtain the running time claimed in Theorem 1.2, we actually need a slightly modified version of FAST-SCHEME that calls IMPROVE repeatedly (see Section 5.)

Definition 2.5. Ranking π of S *respects* partition L, R of S if $\pi(u) < \pi(v)$ for every $u \in L$ and $v \in R$.

The keystone of our analysis is the following Lemma.

LEMMA 2.6. *Let $\beta \geq 0$ and π^{loc} be an ranking such that no single vertex move can improve the cost by more than β. Let k be an integer chosen uniformly at random from $[|S|/3, 2|S|/3]$, L be the set of vertices such that $\pi^{loc}(v) \leq k$ and $R = S \setminus L$. Let π^* be an optimum ranking and π' be an optimum ranking that respects L, R. We have*

$$\mathbf{E}\left[C(\pi')\right] \leq C(\pi^*) + \frac{400}{|S|}\left(\frac{C(\pi^{loc})}{b}\right)^{3/2} + \beta|S|$$

2.3 Why FAST-SCHEME needs single vertex moves

As a warm-up to build intuition we demonstrate that FAST-SCHEME needs single vertex moves. Consider a variant of FAST-SCHEME that calls IMPROVEREC directly on the output of KWIKSORT, without doing any single vertex moves. We now show that this variant is not a PTAS.

Consider a chess tournament (instance) where all the results are consistent except that the worst player beat the best. If KWIKSORT picks any player other than the worst or best as the first pivot it finds a ranking with cost 1, which is optimal. On the other hand, if KWIKSORT picks the best player as the first pivot it produces ranking π^{bad}, which puts the worst player first and has cost $n - 1$.

Now consider what happens when IMPROVEREC is called on π^{bad}. As long as n is not too small IMPROVEREC divides and conquers, irrevocably committing to ranking the worst player among the k best for some $n/3 \leq k \leq 2n/3$. No matter what happens in the later recursions, the output clearly has cost at least $n/3$.

The overall expected cost of the output of this variant of FAST-SCHEME is therefore at least

$$\frac{n-2}{n}1 + \frac{2}{n}\frac{n}{3} = \frac{5}{3} + o(1)$$

hence this variant is not a PTAS.

2.4 Organization of this article

The core of our paper is §3, which proves Lemma 2.6. In §4 we use Lemma 2.6 to prove Lemma 2.4, completing our proof that FAST-SCHEME is an approximation scheme. In §5 we describe a variant of FAST-SCHEME with better runtime and prove Theorem 1.2. In §6 we derandomize. An appendix describes the ADDAPPROX algorithm promised by Theorem 2.3.

3. BOUNDING THE COST OF PARTITION-RESPECTING RANKINGS

3.1 A good partition-respecting ranking

In preparation for proving Lemma 2.6 we adopt its notation. We modify π^* to construct a partition-respecting ranking π^{good} as follows. To avoid having to introduce cumbersome tie-breaking rules, we shift values a little: let $\delta = 1/100$,

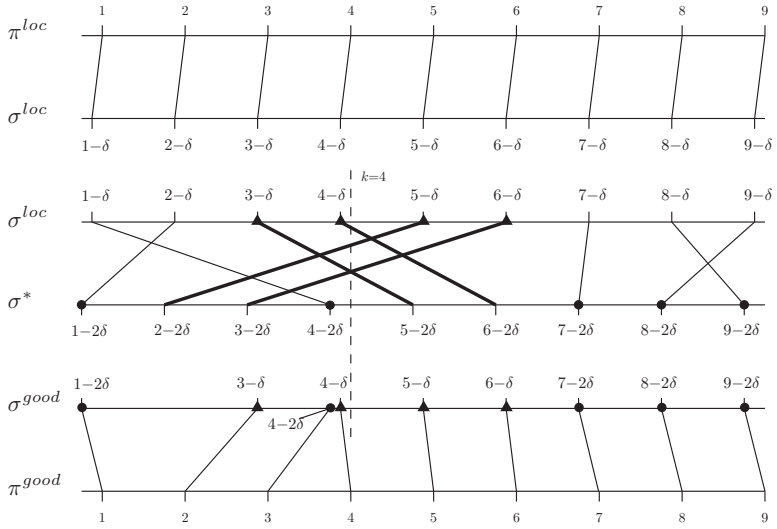

Fig. 3. Definition of π^{good}: edges such that k is between $\sigma^{loc}(u)$ and $\sigma^*(u)$ are in bold.

$\sigma^{loc}(v) = \pi^{loc}(v) - \delta$ and $\sigma^*(v) = \pi^*(v) - 2\delta$. Let

$$\sigma^{good}(v) = \begin{cases} \sigma^*(v) & \text{if } \sigma^{loc}(v) \text{ and } \sigma^*(v) \text{ are on the same side of } k \\ \sigma^{loc}(v) & \text{otherwise} \end{cases}.$$

We then define π^{good} as the ranking naturally associated with σ^{good}: $\pi^{good}(v)$ is the rank of $\sigma^{good}(v)$ among $\{\sigma^{good}(u) : u \in S\}$. See Figure 3 for an illustration.

LEMMA 3.1. *π^{good} respects (L, R).*

PROOF. By definition of σ^{good}, $\sigma^{good}(u) < k$ if and only if $\sigma^{loc}(u) < k$, which holds if and only if $\pi^{loc}(u) \leq k$, or in other words, if and only if $u \in L$. □

3.2 Proof of Lemma 2.6

We call an injective map σ from S to $\mathbb{R}$ an *ordering*. Given ordering σ, we define Ranking(σ) as the ranking naturally associated to σ: Ranking(σ)(v) is the rank of $\sigma(v)$ among $\{\sigma(u) : u \in S\}$. For instance, Ranking(σ^{loc}) = π^{loc}, Ranking(σ^*) = π^*, and Ranking(σ^{good}) = π^{good}. The notion of single vertex move extends naturally to orderings and consists of changing the value of the ordering at a single vertex. We also extend the notion of cost in the obvious way: $C(\sigma) = C(\text{Ranking}(\sigma))$.

For any vertex u, define σ_u^{loc} to be the result of applying a certain single vertex move to σ^{loc}, defined by $\sigma_u^{loc}(v) = \sigma^{loc}(v)$ for all v except for u and $\sigma_u^{loc}(u) = \sigma^*(u)$. For any $x \in \mathbb{R}$ we write *u crosses x* if x is between $\sigma^{loc}(u)$ and $\sigma^*(u)$.

LEMMA 3.2. *Let $T = \sum_{u\,:\,u \text{ crosses } k} \left(C(\sigma^{loc}) - C(\sigma_u^{loc})\right)$. Then $T \leq \beta|S|$.*

PROOF. By definition of π^{loc} (the ranking associated with σ^{loc}), no single vertex move can improve the cost by more than β, so $C(\sigma^{loc}) - C(\sigma_u^{loc}) \leq \beta$. Summing over u concludes the proof. □

LEMMA 3.3. *Let* $F = \sum_{v \in S} |\pi^{loc}(v) - \pi^*(v)|$ *denote the* Spearman's footrule distance *between rankings* π^{loc} *and* π^*. *Then*

$$\mathbf{E}\left[C(\sigma^{good}) - C(\sigma^*) - T\right] \leq \frac{32\sqrt{2}}{|S|} F^{3/2}$$

Before proving Lemma 3.3, let us see how it implies Lemma 2.6.

PROOF. (of Lemma 2.6). By Lemma 3.1 and the definition of π' we have $C(\pi') \leq C(\pi^{good})$, which equals $C(\sigma^{good})$ by the correspondence between rankings and orderings. By Lemma 3.3, in expectation this is at most

$$\mathbf{E}\left[C(\sigma^*) + T\right] + \frac{32\sqrt{2}}{|S|} F^{3/2}.$$

which we can in turn bound using Lemma 3.2 by

$$C(\sigma^*) + \beta|S| + \frac{32\sqrt{2}}{|S|} F^{3/2}$$

By Diaconis and Graham [1977]'s Theorem 2 relating the Spearman's footrule and Kendall-Tau metrics on rankings, we have:

$$\begin{aligned}
F &= \sum_v |\pi^{loc}(v) - \pi^*(v)| \\
&\leq 2 \sum_{u,v} \mathbb{1}\left(\pi^*(u) > \pi^*(v) \text{ and } \pi^{loc}(u) < \pi^{loc}(v)\right) && \text{[D\&G 1977, Thm 2]} \\
&\leq 2 \sum_{u,v} \left[\mathbb{1}\left(\pi^*(u) > \pi^*(v)\right) \frac{w_{uv}}{b} + \mathbb{1}\left(\pi^{loc}(u) < \pi^{loc}(v)\right) \frac{w_{vu}}{b}\right] && \text{since } w_{uv} + w_{vu} \geq b \\
&= (2/b)(C(\pi^*) + C(\pi^{loc})). && \text{by definition of } C \\
&\leq (4/b) C(\pi^{loc}),
\end{aligned}$$

where $\mathbb{1}(p)$ is the indicator function of event p. Therefore

$$\mathbf{E}\left[C(\pi')\right] \leq C(\sigma^*) + \beta|S| + \frac{32\sqrt{2}}{|S|} \left(\frac{4C(\pi^{loc})}{b}\right)^{3/2}.$$

By the correspondence between orderings and rankings again, $C(\sigma^*) = C(\pi^*)$, and clearly $32 \cdot \sqrt{2} \cdot 4^{3/2} < 400$, hence the Lemma. □

3.3 Proof of Lemma 3.3

For any $x, z \in \mathbb{R}$, define $(x, z) = \{\, y \in \mathbb{R} : x < y < z \text{ or } z < y < x \,\}$. We say $\{u, v\}$ is a *crossing pair* if the intervals $\left(\sigma^*(u), \sigma^{loc}(u)\right)$ and $\left(\sigma^*(v), \sigma^{loc}(v)\right)$ intersect but neither is contained within the other. We say that $\{u, v\}$ is a *cut crossing pair* if it is a crossing pair and $k \in \left(\sigma^*(u), \sigma^{loc}(u)\right) \cup \left(\sigma^*(v), \sigma^{loc}(v)\right)$.

LEMMA 3.4.

$$C(\sigma^{good}) - C(\sigma^*) - T \leq 4|\{\{u, v\} \textit{ cut crossing pair}\}|$$

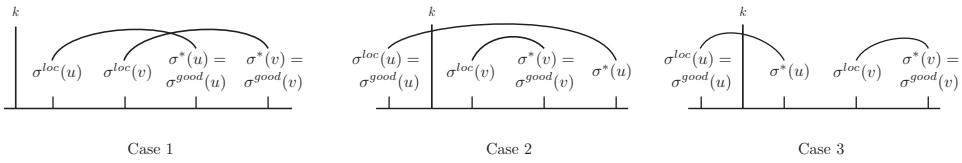

Fig. 4. Illustration of cases in proof of Lemma 3.4

PROOF. We use the following notation: given an ordering σ, let $C(\sigma)_{uv}$ denote the contribution of edge $\{u, v\}$ to the cost of σ, that is, w_{uv} or w_{vu} depending on the relative order of u and v. Let $t_{uv} = \mathbb{1}\left(u \text{ crosses } k\right)\left(C(\sigma^{loc})_{uv} - C(\sigma^{loc}_u)_{uv}\right)$. With these notations, it trivially follows that

$$C(\sigma^{good}) - C(\sigma^*) - T = \sum_{\{u,v\}} \left[C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} - (t_{uv} + t_{vu})\right].$$

If $\{u, v\}$ is a cut-crossing pair, we use naïve bounds: $C(\sigma^{good})_{uv}, C(\sigma^*)_{uv}, t_{uv}$ and t_{vu} are all at most 1 in absolute value, and so

$$C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} - (t_{uv} + t_{vu}) \leq 4.$$

If $\{u, v\}$ is not a cut crossing pair, we do a case-by-case analysis to prove that

$$C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} - (t_{uv} + t_{vu}) = 0.$$

The three cases are illustrated in Figure 4.

Case 1: If neither u nor v cross k, then $C(\sigma^{good})_{uv} = C(\sigma^*)_{uv}$ by definition of σ^{good}, and $t_{uv} = t_{vu} = 0$ by definition of t.

Case 2: Suppose either v or u (or both) cross k and the intervals $\left(\sigma^*(u), \sigma^{loc}(u)\right)$ and $\left(\sigma^*(v), \sigma^{loc}(v)\right)$ are nested. Without loss of generality suppose $\left(\sigma^*(v), \sigma^{loc}(v)\right)$ is contained within $\left(\sigma^*(u), \sigma^{loc}(u)\right)$. It is impossible for v to cross k without u crossing k as well so we conclude u crosses k. Whether the image of v is $\sigma^{loc}(v)$ or $\sigma^*(v)$ does not affect the orientation of edge uv, and so $C(\sigma^{good})_{uv} = C(\sigma^{loc})_{uv}$, $C(\sigma^*)_{uv} = C(\sigma^{loc}_u)_{uv}$ and $C(\sigma^{loc}_v)_{uv} = C(\sigma^{loc})_{uv}$. Therefore by definitions $C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} = t_{uv}$ and $t_{vu} = 0$.

Case 3: Suppose either v or u (or both) cross k and the intervals $\left(\sigma^*(u), \sigma^{loc}(u)\right)$ and $\left(\sigma^*(v), \sigma^{loc}(v)\right)$ are disjoint. The edge uv is oriented the same way in all of the relevant orderings so $C(\sigma^{good})_{uv} = C(\sigma^{loc})_{uv} = C(\sigma^*)_{uv} = C(\sigma^{loc}_u)_{uv} = C(\sigma^{loc}_v)_{uv}$. Therefore $C(\sigma^{good})_{uv} - C(\sigma^*)_{uv} = t_{uv} = t_{vu} = 0$.

This proves the Lemma. □

LEMMA 3.5. *For any $u \in S$, the probability that u crosses k is at most*

$$\frac{4|\pi^{loc}(u) - \pi^*(u)|}{|S|}.$$

PROOF. Let $K = \{\, k \;:\; |S|/3 \leq k \leq 2|S|/3 \,\}$ be the set k is chosen randomly from. The probability that u crosses k equals $|\{\, k \in K \;:\; u \text{ crosses } k \,\}|/|K|$. The numerator $|\{\, k \in K \;:\; u \text{ crosses } k \,\}|$ is at most $|\pi^{loc}(u) - \pi^*(u)|$. The denominator $|K|$ is approximately $|S|/3$, and a careful analysis for $|S|$ small shows that for $|S| > 1$ it is always at least $|S|/4$, hence the result. □

LEMMA 3.6. *For any vertex $u \in S$, $|\{\, v \in S \,:\, \{u,v\}$ crossing pair$\,\}| \leq 2\sqrt{2F}$*

PROOF. Fix a vertex u. Observe that if $\{u,v\}$ is a crossing pair, then v must cross $\lceil \sigma^{loc}(u) \rceil$ or $\lfloor \sigma^*(u) \rfloor$ (the floor and ceiling come from the shift by $-\delta$ for σ^{loc} and by -2δ for σ^*). Thus

$$|\{\, v \in S \,:\, \{u,v\} \text{ crossing pair}\,\}| \;\leq\; \xi(\lceil \sigma^{loc}(u) \rceil) + \xi(\lfloor \sigma^*(u) \rfloor), \tag{1}$$

where $\xi(j)$ denotes the number of vertices that cross integer j.

Now we need to relate $\xi(i)$ to the footrule distance F. We write:

$$F \;=\; \sum_u |\pi^{loc}(u) - \pi^*(u)| = \sum_u \sum_{j \in \mathbb{Z}} \mathbb{1}\left(j \in \left(\sigma^*(u), \sigma^{loc}(u)\right)\right) = \sum_{j \in \mathbb{Z}} \xi(j).$$

Now, it is easy to see that we always have $|\xi(j) - \xi(j+1)| \leq 2$ (since the difference can only come from two vertices: the vertex such that $\sigma^{loc}(u) = j + 1 - \delta$, and the vertex such that $\sigma^*(u) = j + 1 - \delta$.) Thus for any i we have:

$$\sum_{j \in \mathbb{Z}} \xi(j) \geq \xi(i) + 2 \sum_{j=1}^{\lfloor \xi(i)/2 \rfloor} (\xi(i) - 2j) \geq \frac{\xi(i)^2}{2},$$

where the last inequality follows after a straightforward calculation. Thus for all i

$$\xi(i) \leq \sqrt{2F} \tag{2}$$

Applying Equation (2) to $i = \lceil \sigma^{loc}(u) \rceil$ and to $i = \lfloor \sigma^*(u) \rfloor$, summing the results and plugging into Equation (1) yields the Lemma. □

Now we prove Lemma 3.3.

PROOF. By Lemmas 3.4, 3.5 and 3.6

$$\begin{aligned}
&\mathbf{E}\left[C(\sigma^{good}) - C(\sigma^*) - T\right] \\
&\leq\; 4 \sum_u \Pr\left(u \text{ crosses } k\right) \cdot |\{\, v \in S \,:\, \{u,v\} \text{ crossing pair}\,\}| \\
&\leq\; 4 \sum_u \frac{4|\pi^{loc}(u) - \pi^*(u)|}{|S|} 2\sqrt{2F} = 32\sqrt{2}\frac{F^{3/2}}{|S|}.
\end{aligned}$$

□

4. SUMMING ERRORS OVER THE RECURSION TREE (PROOF OF LEMMA 2.4)

Consider each set S on which we execute algorithm IMPROVEREC, and call such a set *leaf* or *internal* depending on whether or not there are further recursive calls to IMPROVEREC. Let π_S^{loc} and π_S^{out} denote respectively the input and output orderings of IMPROVEREC when run on S. Let π' denote the optimal ranking of S that respects (L,R), π'_L denote the optimal ranking of L (which is also the restriction of π' to L) and π'_R denote the optimal ranking of R (which is also the restriction of π' to R). Let π_L^{out} and π_R^{out} denote the restrictions of π^{out} to L and R respectively. The key observation is:

$$\begin{aligned}
&\mathbf{E}\left[C(\pi_S^{out})\right] - OPT_S \\
&=\; \mathbf{E}\left[C(\pi_S^{out}) - C(\pi')\right] + \mathbf{E}\left[C(\pi') - OPT_S\right] \\
&=\; \mathbf{E}\left[C(\pi_L^{out}) - C(\pi'_L)\right] + \mathbf{E}\left[C(\pi_R^{out}) - C(\pi'_R)\right] + \mathbf{E}\left[C(\pi') - OPT_S\right]
\end{aligned} \tag{3}$$

because π^{out} and π' both respect L, R. To be more precise about the meanings of the expectations in(3) let $\mathbb{T}_S$ be a random variable expressing the random choices made by ImproveRec(S, π_S^{loc}) and all of its descendants. We restate (3) formally as:

$$\begin{aligned}&\mathbf{E}_{\mathbb{T}_S}\left[C(\pi_S^{out})\right] - OPT_S \\ &= \mathbf{E}_k\left[\mathbf{E}_{\mathbb{T}_L}\left[C(\pi_L^{out}) - OPT_L\right]\right] + \mathbf{E}_k\left[\mathbf{E}_{\mathbb{T}_R}\left[C(\pi_R^{out}) - OPT_R\right]\right] + \\ &\quad + \mathbf{E}_k\left[C(\pi') - OPT_S\right] \end{aligned} \tag{4}$$

Now use Lemma 2.6 to write

$$\mathbf{E}_k\left[C(\pi') - OPT_S\right] \le \frac{400}{|S|}\left(\frac{C(\pi^{loc})}{b}\right)^{3/2} + \beta|S|. \tag{5}$$

Applying (4) and (5) repeatedly over the recursion tree we get

$$\begin{aligned}&\mathbf{E}_{\mathbb{T}_V}\left[C(\pi_V^{out})\right] - OPT \\ &\le \mathbf{E}_{\mathbb{T}_V}\left[\sum_{S \text{ leaf}} \mathbf{E}_{\mathbb{T}_S}\left[C(\pi_S^{out}) - OPT_S\right] + \sum_{S \text{ internal}} \beta|S| + \sum_{S \text{ internal}} \frac{400}{|S|}\left(\frac{C(\pi_S^{loc})}{b}\right)^{3/2}\right].\end{aligned} \tag{6}$$

To complete the proof we bound the argument of the expectation that is the right hand side of (6) by $\eta C(\pi_V^{loc})$ uniformly over any set of random choices $\mathbb{T}_V$. Dealing with the first sum is straightforward: any leaf must have $\mathbf{E}\left[C(\pi_S^{out})\right] \le OPT_S + \frac{\eta}{4}\kappa|S|^2 \le OPT_S + \frac{\eta}{4}C(\pi_S^{loc})$. Summing,

$$\sum_{S \text{ leaf}} \mathbf{E}\left[C(\pi_S^{out}) - OPT_S\right] \le \frac{\eta}{4}\sum_{S \text{ leaf}} C(\pi_S^{loc}) \le \frac{\eta}{4}C(\pi_V^{loc}) \le \frac{\eta}{4}C(\pi_V^{in})$$

where π^{in} is the ranking input to Improve.

Dealing with the second sum in (6) is also straightforward: note that the tree of recursive calls has at most $\log_{3/2} n$ levels of internal nodes and that on each level the sets S are disjoint so

$$\sum_{S \text{ internal}} \beta|S| \le \sum_{\text{levels}} \beta n \le \beta n \log_{3/2} n \le \frac{\eta}{8}C(\pi_V^{in}).$$

To deal with the third sum in (6), we start with two preliminary lemmas. Consider an internal node S and its two children L and R.

Lemma 4.1. *Let c_S, c_L, c_R denote $C(\pi_S^{loc})$, $C(\pi_L^{loc})$ and $C(\pi_R^{loc})$ respectively. Then:*

$$\frac{(c_S - c_L)^{3/2}}{|S| - |L|} - \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R) \le \frac{c_R^{3/2}}{|R|} \le \frac{(c_S - c_L)^{3/2}}{|S| - |L|}.$$

Proof. Let $\alpha = \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}$. The second inequality is obvious since $|R| = |S| - |L|$ and $c_S \ge c_L + c_R$. To prove the first inequality, let $\phi(x) = x^{3/2}/|R| - \alpha x$. Note that the derivative $\phi'(x) = (3/2)(x^{1/2}/|R|) - \alpha$ is increasing. Since $|R| \ge |S|/3$, we have $\phi'(c_S) \le (1/2)(c_S^{1/2}/|S|) - \alpha$. Since S is internal, by definition of the algorithm $c_S \le \kappa|S|^2$, which implies $\phi'(c_S) \le 0$ by definition of κ and α. So, ϕ is decreasing in the range $[0, c_S]$. Since $c_R \le c_S - c_L \le c_S$, we deduce $\phi(c_R) \ge \phi(c_S - c_L)$, hence the lemma. □

LEMMA 4.2. *For any $x \in [0,1]$ and $y \in [1/3, 2/3]$, we have:*

$$\frac{x^{3/2}}{y} + \frac{(1-x)^{3/2}}{1-y} \geq 4/3.$$

PROOF. We study the minima of $f(x,y) = x^{3/2}/y + (1-x)^{3/2}/(1-y)$.

If x is at the boundary, say $x = 0$, then $f(x,y) = 1/(1-y)$ which has minimum $3/2$ for the specified range of y. The case $x = 1$ is symmetric. If x is not at a boundary then any local minimum must satisfy $\frac{\partial f}{\partial x} = 0$, hence $\frac{\sqrt{x}}{y} = \frac{\sqrt{1-x}}{1-y}$. If y is at the boundary, say $y = 1/3$, this gives $x = 1/5$, hence $f(x,y) = 3/\sqrt{5}$. The case $y = 2/3$ is symmetric. Finally, if (x,y) is in the interior of the domain then we must have $\frac{\partial f}{\partial y} = 0$ as well, hence $x = y = 1/2$, and $f(x,y) = \sqrt{2}$. Finally the minimum of those three values is $3/\sqrt{5} > 4/3$. □

Applying first Lemma 4.1 and then Lemma 4.2 for $x = c_L/c_S$ and $y = |L|/|S|$ yields

$$\begin{aligned}\frac{c_L^{3/2}}{|L|} + \frac{c_R^{3/2}}{|R|} &\geq \frac{c_L^{3/2}}{|L|} + \frac{(c_S - c_L)^{3/2}}{|S| - |L|} - \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R) \\ &\geq (4/3)\frac{c_S^{3/2}}{|S|} - \frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R)\end{aligned}$$

Thus,

$$\frac{c_S^{3/2}}{|S|} \leq (3/4)\frac{c_L^{3/2}}{|L|} + (3/4)\frac{c_R^{3/2}}{|R|} + \frac{3}{4}\frac{b^{3/2}}{400}\frac{3}{2}\frac{\eta}{27}(c_S - c_L - c_R). \tag{7}$$

The rest of the analysis is straightforward. Applying Equation (7) from the root down, we bound the third sum of (6), call it A, as follows:

$$\begin{aligned}A &= \sum_{S \text{ internal}} \frac{400}{b^{3/2}}\frac{C(\pi_S^{loc})^{3/2}}{|S|} \\ &\leq \left(\sum_{S \text{ leaf}} \frac{400}{b^{3/2}}\frac{C(\pi_S^{loc})^{3/2}}{|S|}\sum_{i \geq 1}(3/4)^i\right) + \frac{3}{4}\frac{1}{400}\frac{3}{2}\frac{\eta}{27}C(\pi_V^{loc}).\end{aligned} \tag{8}$$

We bound the second term of (8) trivially by $(\eta/8)C(\pi_V^{in})$.

Let S be a leaf and S' its parent.[2] Since $C(\pi_S^{loc}) \leq C(\pi_{S'}^{loc})$ and $|S| \geq |S'|/3$, we have

$$\frac{C(\pi_S^{loc})^{1/2}}{|S|} \leq 3\frac{C(\pi_{S'}^{loc})^{1/2}}{|S'|} \leq 3\sqrt{\kappa} = 3\sqrt{\frac{\eta^2 b^3}{350 \cdot 400^2}} \leq \frac{3\eta b^{3/2}}{18 \cdot 400}$$

by definition of internal nodes. Substituting and using the fact that leaves are disjoint, we bound the first term of (8) by

$$\sum_{S \text{ leaf}} \frac{400}{b^{3/2}}\frac{3\eta b^{3/2}}{18 \cdot 400}\frac{3/4}{1 - 3/4}C(\pi_S^{loc}) = \frac{\eta}{2}C(\pi_V^{loc}) \leq \frac{\eta}{2}C(\pi_V^{in}).$$

[2]If the root is a leaf no such parent exists, but in that case $A = 0$.

FASTer-Scheme:

Run KwikSort to define an ranking π
for $i \leftarrow 1$ to $\lceil \log_2 1/b \rceil$ **do**
  $\pi \leftarrow$ Improve$(\pi, 1/2)$
end for
Return Improve$(\pi, \epsilon/7)$.

Fig. 5. Faster approximation scheme (error tolerance $\epsilon > 0$)

Therefore $A \leq (1/8 + 1/2)C(\pi_V^{in}) = 5/8C(\pi_V^{in})$, completing the proof of the lemma.

5. RUNTIME

5.1 A faster algorithm

To speed up FAST-Scheme, we perform the following two modifications: First, instead of calling Improve just once with error tolerance $\epsilon b/5$, we first call it $\log(1/b)$ times with error tolerance $1/2$ before running it once with error tolerance $\epsilon/7$. The improved approximation scheme, denoted FASTer-Scheme, is presented in Figure 5. Second, to bound the total runtime of the single-vertex moves, we need the cost to be monotone non-increasing. For that purpose, we modify algorithm Improve, replacing the line

Return ImproveRec(V, π)

with

Return either π or ImproveRec(V, π), whichever has lower cost.

Lemma 2.4 clearly remains valid despite this modification.

We now prove that FASTer-Scheme is a $1 + \epsilon$-approximation.

Proof. (Of Theorem 1.2) Let $m = \lceil \log_2 1/b \rceil$ and π^i denote the ranking π after the i^{th} iteration of FASTer-Scheme, $0 \leq i \leq m$. By the law of iterated expectations and Lemma 2.4, for any $1 \leq i \leq m$ we have

$$\mathbf{E}\left[C(\pi^i)\right] = \mathbf{E}\left[\mathbf{E}\left[C(\pi^i)|\pi^{i-1}\right]\right] \leq \mathbf{E}\left[OPT + \frac{1}{2}C(\pi^{i-1})\right] = OPT + \frac{1}{2}\mathbf{E}\left[C(\pi^{i-1})\right].$$

Therefore by Corollary 2.2 and the definition of m,

$$\mathbf{E}\left[C(\pi^m)\right] \leq 2OPT + 2^{-m}\frac{5}{b}OPT \leq 7OPT. \tag{9}$$

Finally, by Lemma 2.4, the expected cost of the output is at most

$$OPT + \frac{\epsilon}{7}\mathbf{E}\left[C(\pi^m)\right] = (1+\epsilon)OPT.$$

□

5.2 Analysis of running time

Here we analyze the runtime of FASTer-Scheme. Throughout this section let $n = |V|$, the number of vertices in the overall input.

LEMMA 5.1. *All the single vertex move local optimizations in* FASTER-SCHEME *have a combined expected runtime of* $O(n^3 \log n(\log(1/b) + \frac{1}{\epsilon}))$.

PROOF. The modification to IMPROVE discussed in subsection 5.1 ensures that after a single vertex move yields a ranking with cost C, FASTER-SCHEME will never apply a single vertex move to an ranking costing at least C. Let π^0 denote the constant-factor ordering in FASTER-SCHEME. Each single vertex move multiplies the cost of the ranking by a factor of at most $1 - \frac{\eta}{4n \log_{3/2} n}$. Therefore after j moves, $C(\pi) \leq C(\pi^0)(1 - \frac{\eta}{4n \log_{3/2} n})^j \leq C(\pi^0) e^{-j \frac{\eta}{4n \log_{3/2} n}}$. Clearly $C(\pi) \geq OPT$ hence $\mathbf{E}[j]$ is at most $\mathbf{E}\left[\frac{4n \log_{3/2} n}{\eta} \ln \frac{OPT}{C(\pi^0)}\right] \leq \frac{4n \log_{3/2} n}{\eta} \ln \mathbf{E}\left[\frac{C(\pi^0)}{OPT}\right] \leq \frac{4n \log_{3/2} n}{\eta} \ln(5/b)$ moves are needed in expectation. During all the calls to IMPROVE except the last $\eta = 1/2$, hence at most $\frac{4n \log_{3/2} n}{1/2} \ln \frac{5}{b} = O(n \log n \log(1/b))$ moves are required. During the final call to IMPROVE $\eta = \epsilon/7$ hence using a similar argument and (9) at most $\frac{4n \log_{3/2} n}{\epsilon/7} \ln \mathbf{E}\left[\frac{C(\pi^m)}{OPT}\right] = O\left(\frac{n \log_{3/2} n}{\epsilon}\right)$ moves are required. There are an additional $\lceil \log(1/b) \rceil + 1$ times that no improving move exists but this fact must still be verified. Overall one must check for local optimality and make an improving move if one exists $O(n \log n \log(1/b)) + O(\frac{n \log n}{\epsilon}) + O(\log 1/b) = O(n \log n(\log(1/b) + \frac{1}{\epsilon}))$ times. Each check for local optimality can be trivially done in $O(n^2)$ time, so the single vertex moves take $O(n^3 \log n(\log(1/b) + \frac{1}{\epsilon}))$ time overall. □

LEMMA 5.2. IMPROVEREC *has runtime* $O(n^2) + n2^{\tilde{O}(1/\eta^6)}$.

PROOF. We refer to the call to IMPROVEREC made by IMPROVE the *root call*. We first analyze the runtime of the ADDAPPROX calls. Recall from Theorem 2.3 that each ADDAPPROX call has runtime $O(|S|^2) + 2^{O(1/\eta^6)}$. Each vertex participates in exactly one ADDAPPROX call, so the total runtime of the ADDAPPROX calls descended from the root call is $O(n^2) + n2^{\tilde{O}(1/\eta^6)}$.

Each call to IMPROVEREC, excluding descendent IMPROVEREC and ADDAPPROX calls, can easily be implemented to run in $O(|S|^2)$ time. Summing over the levels of the recursion tree gives total runtime $O\left(n^2 + n\frac{2n}{3} + n\frac{2^2 n}{3^2} + \ldots\right) = O\left(n^2\right)$. □

We now prove the runtime portion of Theorem 1.2.

PROOF. The overall runtime of FASTER-SCHEME is:

$$
\begin{aligned}
&O(n \log n) && \text{(KWIKSORT [Ailon and Mohri 2008])}\\
&+ O(n^3 \log n(\log(1/b) + \tfrac{1}{\epsilon})) && \text{(Single vertex moves, Lemma 5.1)}\\
&+ O(\log(1/b)) \left(O(n^2) + n2^{\tilde{O}(1/b^6)}\right) && \text{(IMPROVEREC } \eta = 1/2\text{, Lemma 5.2)}\\
&+ \left(O(n^2) + n2^{\tilde{O}(1/(\epsilon b)^6)}\right) && \text{(IMPROVEREC } \eta = \epsilon/7\text{, Lemma 5.2)}\\
&= O\left(n^3 \log n(\log(\tfrac{1}{b}) + \tfrac{1}{\epsilon})\right) + n\left(2^{\tilde{O}(1/(\epsilon b)^6)}\right)
\end{aligned}
$$

□

6. DERANDOMIZATION

The KWIKSORT algorithm was derandomized in [Van Zuylen et al. 2007; Van Zuylen and Williamson 2007]. The following theorem is implicit in the proof of

ImproveRec(vertices S, ranking π on S):

if ImproveRec(S, π) previously computed **then**
  Return cached π^{out}.
else if $|S| = 1$ **then**
  Return π
else if $C(\pi) \geq \kappa |S|^2$ **then**
  Return the ranking from the deterministic additive error algorithm
  by Frieze and Kannan [1999] with $\delta = \frac{\eta}{4} \cdot \kappa$
else
  Initialize π^{out} to an arbitrary ranking (e.g. π).
  for $k = \lceil |S|/3 \rceil$ to $\lfloor 2|S|/3 \rfloor$ **do**
    Let $L = \{ v \in S : \pi(v) \leq k \}$ and $R = S \setminus L$
    Let π_L be the ranking of L induced by π, i.e. $\pi_L(v) = \pi(v)$
    Let π_R be the ranking of R induced by π, i.e. $\pi_R(v) = \pi(v) - k$
    Let π^{temp} be the concat. of ImproveRec(L, π_L) and ImproveRec(R, π_R).
    if $C(\pi^{temp}) < C(\pi^{out})$ **then**
      $\pi^{out} \leftarrow \pi^{temp}$
    end if
  end for
  Return π^{out}.
end if

Fig. 6. Derandomized version of ImproveRec.

Theorem 2.1 in Van Zuylen and Williamson [2007]:

Theorem 6.1. *[Van Zuylen and Williamson 2007] Let w be non-negative weight function on the edges. Assume that for every $u, v, x \in V$ with $w_{vu} - w_{uv} \leq \min\{w_{xv} - w_{vx}, w_{xu} - w_{ux}\}$ we have*

$$w_{vu} + \frac{1}{2}(w_{vx} + w_{xv}) + \frac{1}{2}(w_{xu} + w_{ux}) \geq \frac{1}{\alpha}(w_{uv} + w_{vx} + w_{xu})$$

for some $\alpha > 1$. Then there exists a deterministic polynomial-time α-approximation algorithm.

Theorem 6.2. *There exists a deterministic $3/b$-approximation algorithm for weighted feedback act set tournament.*

Proof. (Adapted from the proof by Van Zuylen and Williamson [2007] for the case $b = 1$.) We know $w_{vx} + w_{xv}, w_{xu} + w_{ux} \geq b$ and $w \leq 1$ hence

$$w_{vu} + \frac{1}{2}(w_{vx} + w_{xv}) + \frac{1}{2}(w_{xu} + w_{ux}) \geq 0 + \frac{b}{2} + \frac{b}{2} = b = \frac{1}{3/b} 3 \geq \frac{1}{3/b}(w_{uv} + w_{vx} + w_{xu})$$

□

To derandomize FAST-Scheme we make three changes to our algorithms. Firstly, we replace AddApprox with the deterministic additive error algorithm by Frieze and Kannan [1999]. Secondly, we replace KwikSort with the deterministic constant-factor approximation algorithm of Theorem 6.2.

Thirdly we must eliminate the randomized choice of k in ImproveRec. We do this by trying every possible k and keeping the best ranking found. We cache intermediate results to prevent exponential blow-up in the runtime. The derandomized version of ImproveRec is given in Figure 6.

AddApprox:

Take a sample S of $\tilde{O}(1/\delta^4)$ vertices chosen uniformly at random without replacement from V.
Take a sample T of $\tilde{O}(1/\delta^2)$ vertices chosen uniformly at random without replacement from S.
for each of the possible bucketed rankings of T **do**
  for each vertex v of $S \setminus T$ in random order **do**
    Place v in the bucket that minimizes the cost (of the bucketed ranking of the vertices placed so far.)
  end for
end for
Extend the discovered bucketed ranking of S with the minimum cost as follows:
for each vertex v of $V \setminus S$ in random order **do**
  Place v in the bucket that minimizes the cost so far.
end for

Fig. 7. Algorithm for bucketed FAS with additive error $O(\delta n^2)$

Let π^{loc} denote the input order. It is easy to see that every (S, π) pair encountered is of the form $S = \{ u : i \leq \pi^{loc}(u) \leq j \}$ and $\pi(u) = \pi^{loc}(v) - i$ for some $i \leq j$, so ImproveRec runs $O(n^2)$ times (excluding lookups in cache). Each call to ImproveRec has runtime dominated by the derandomized additive error algorithm, with runtime $n^{\tilde{O}(1/((\epsilon b)^3)^4)} = n^{\tilde{O}(1/(\epsilon b)^{12})}$. The overall runtime of Improve is therefore $O\left(n^2\right) \cdot n^{\tilde{O}(1/(\epsilon b)^{12})} = n^{\tilde{O}(1/(\epsilon b)^{12})}$.

The runtime of the derandomized FAST-Scheme is dominated by the $O(\log 1/b)$ calls to Improve so the overall runtime is $(\log 1/b) n^{\tilde{O}(1/(\epsilon b)^{12})} = n^{\tilde{O}(1/(\epsilon b)^{12})}$.

A. APPENDIX

In this appendix we prove Theorem 2.3. We want to find a $O(\delta n^2)$ additive approximation in time $O(n^2) + 2^{\tilde{O}(1/\delta^2)}$ for any constant $\delta > 0$.

Let a *bucketed ranking* be a function $\tilde{\pi}$ from V to $\{1, 2, 3, \ldots \lceil \frac{1}{\delta} \rceil\}$. If $\tilde{\pi}(v) = i$ we say that vertex v is in bucket i.

Problem A.1 *Bucketed FAS.*
Input: complete directed graph with vertex set V, $n \equiv |V|$ and non-negative edge weights w_{uv} with $w_{uv} + w_{vu} \leq 1$.
Output: A bucketed ranking minimizing $\tilde{C}(\tilde{\pi}) = \sum_{\{u,v\} \subset V : \tilde{\pi}(v) \geq \tilde{\pi}(u)} w_{vu}$.

Note the "$\geq$" in the objective function $\tilde{C}$ of Problem A.1 – if vertices u and v are in the same bucket then $\tilde{C}$ pays for both w_{uv} and w_{vu}.

Algorithm 7 is the specialization of an algorithm by Mathieu and Schudy [2008] to the bucketed feedback arc set problem. We remark that in Algorithm 7 $|T| = \tilde{O}(1/\delta^2)$ whereas Mathieu and Schudy [2008] has $|T| = O(1/\delta^2)$ because Mathieu and Schudy [2008] assume that the variables have constant-sized domains whereas we have a number of buckets that depends on δ.

Theorem A.2. *[Mathieu and Schudy 2008] Algorithm 7 has expected additive error $O(\delta n^2)$.*

Now we prove Theorem 2.3.

PROOF. Run Algorithm 7 and convert the output bucketed ranking $\widetilde{OUT}$ into a ranking OUT by concatenating the buckets and ordering arbitrarily within each bucket. The pessimistic definition of $\tilde{C}$ ensures $C(OUT) \leq \tilde{C}(\widetilde{OUT})$. On the other hand, the optimum ranking OPT can be converted into a bucketed ranking $\widetilde{OPT}$ that costs at most $O(\delta n^2)$ more by placing δn vertices in each bucket. Putting these remarks together with Theorem A.2 we see

$$C(OUT) \;\leq\; \tilde{C}(\widetilde{OUT}) \leq \tilde{C}(\widetilde{OPT}) + O(\delta n^2) \leq C(OPT) + O(\delta n^2) + O(\delta n^2)$$

proving the approximation factor portion of Theorem 2.3.

Determining where to place each vertex can be done in time $O(m/\delta)$ totally naïvely, where m is the number of vertices placed so far. Exploiting the specific structure of $\tilde{C}$ allows each greedy choice to be made in time $O(m + \frac{1}{\delta})$, which yields the runtime $O(n^2) + 2^{\tilde{O}(1/\delta^2)}$ of Theorem 2.3. □

ACKNOWLEDGMENTS

We would like to that Nir Ailon, Marek Karpinski and Eli Upfal for helpful discussion and the reviewers of the conference version of this work for useful suggestions.

REFERENCES

AILON, N. 2007. Aggregation of partial rankings, p-ratings and top-m lists. In *Procs. 18th ACM-SIAM SODA*. 415–424. Journal version to appear in *Algorithmica*.

AILON, N., CHARIKAR, M., AND NEWMAN, A. 2008. Aggregating inconsistent information: ranking and clustering. *J. ACM 55,* 5, Article 23.

AILON, N. AND MOHRI, M. 2008. An efficient reduction of ranking to classification. In *Procs. 21st COLT*. 87–97.

ALON, N. 2006. Ranking tournaments. *SIAM J. Discrete Math. 20,* 1, 137–142.

ARORA, S., FRIEZE, A. M., AND KAPLAN, H. 2002. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical Programming 92,* 1, 1–36.

BAKER, F. AND HUBERT, L. 1977. Applications of combinatorial programming to data analysis: seriation using symmetric proximity measures. *British J. Math. Statis. Psych. 30*, 154–164.

BARTHOLDI, III, J., TOVEY, C., AND TRICK, M. 1989. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare 6*, 157–165.

BERGER, B. AND SHOR, P. 1997. Tight bounds for the maximum acyclic subgraph problem. *J. Algorithms 25,* 1, 1–18.

BETZLER, N., FELLOWS, M. R., GUO, J., NIEDERMEIER, R., AND ROSAMOND, F. A. 2008. Fixed-parameter algorithms for Kemeny scores. In *AAIM '08: Procs. 4th int'l conf. on Algorithmic Aspects in Information and Management*. 60–71.

BORDA, J. 1781. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*.

BRAVERMAN, M. AND MOSSEL, E. 2008. Noisy sorting without resampling. In *Procs. 19th ACM-SIAM SODA*. 268–276.

CAI, M., DENG, X., AND ZANG, W. 2001. An approximation algorithm for feedback vertex sets in tournaments. *SIAM J. Computing 30,* 6, 1993–2007.

CHARBIT, P., THOMASSE, S., AND YEO, A. 2007. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability and Computing 16*, 1–4.

CHARIKAR, M., GURUSWAMI, V., AND WIRTH, A. 2005. Clustering with qualitative information. *J. Computer and System Sciences 71,* 3, 360 – 383.

CHARIKAR, M., MAKARYCHEV, K., AND MAKARYCHEV, Y. 2007a. A divide and conquer algorithm for d-dimensional arrangement. In *Procs. 18th ACM-SIAM SODA*. 541–546.

Charikar, M., Makarychev, K., and Makarychev, Y. 2007b. On the advantage over random for maximum acyclic subgraph. In *Procs. 48^{th} IEEE FOCS*. 625–633.

Cohen, W. W., Schapire, R. E., and Singer, Y. 1999. Learning to order things. *J. Artificial Intelligence Research 10*, 243–270.

Coleman, T. and Wirth, A. 2008. Ranking tournaments: local search and a new algorithm. In *Procs. 10^{th} workshop on Algorithm Engineering and Experiments (ALENEX)*. 133–141.

Condorcet, M. J. 1785. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Reprinted by AMS Bookstore in 1972.

Conitzer, V. 2006. Computer Slater rankings using similarities among candidates. In *Procs. 21^{st} AAAI*. 613–619.

Conitzer, V., Davenport, A., and Kalagnanam, J. 2006. Improved bounds for computing Kemeny rankings. In *Proc. 21^{st} AAAI*. 620–626.

Coppersmith, D., Fleischer, L., and Rudra, A. 2006. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Procs. 17^{th} ACM-SIAM SODA*. 776–782.

Demetrescu, C. and Finocchi, I. 2000. Break the "right" cycles and get the "best" drawing. In *Procs. 2^{nd} Int'l workshop on Algorithmic Engineering and Experiments (ALENEX)*. 171–182.

Demetrescu, C. and Finocchi, I. 2001. Breaking cycles for minimizing crossings. *J. Experimental Algorithmics 6*, Article 2.

Diaconis, P. and Graham, R. 1977. Spearman's footrule as a measure of disarray. *J. Royal Statistical Society 39,* 2, 262–268.

Dinur, I. and Safra, S. 2002. The importance of being biased. In *Procs. 34^{th} ACM STOC*. 33–42.

Dom, M., Guo, J., Hüffner, F., Niedermeier, R., and Truss, A. 2006. Fixed-parameter tractability results for feedback set problems in tournaments. In *LNCS*. Vol. 3998. Springer, 320–331.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. 2001. Rank aggegation methods for the web. In *Procs. 10^{th} WWW*. 613–622. The NP-hardness proof is in the online-only appendix available from `http://www10.org/cdrom/papers/577/`.

Eades, P., Lin, X., and Smyth, W. 1993. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters 47,* 6, 319–323.

Erdös, P. and Moon, J. 1965. On sets of consistent arcs in tournaments. *Canadian Math. Bulliten 8,* 3, 269–271.

Even, G., Naor, J., Schieber, B., and Sudan, M. 1998. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica 20,* 2, 151–174.

Even, G., Naor, J. S., Rao, S., and Schieber, B. 2000. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM 47,* 4, 585–616.

Fernandez de la Vega, W. 1983. On the maximal cardinality of a consistent set of arcs in a random tournament. *J. Combinatorial Theory, Ser. B 35,* 3, 328–332.

Flood, M. 1990. Exact and heuristic algorithms for the weighted feedback arc set problem: a special case of the skew-symmetric quadratic assignment problem. *Networks 20,* 1, 1–23.

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

Grötschel, M., Jünger, M., and Reinelt, G. 1985a. Facets of the linear ordering polytope. *Math Programming 33*, 43–60.

Grötschel, M., Jünger, M., and Reinelt, G. 1985b. On the acyclic subgraph polytope. *Math Programming 33*, 28–42.

Hassin, R. and Rubinstein, S. 1994. Approximations for the maximum acyclic subgraph problem. *Information Processing Letters 51,* 3, 133–140.

Jung, H. A. 1970. On subgraphs without cycles in tournaments. In *Combinatorial theory and its applications II*, P. Erdös, A. Rényi, and V. Sós, Eds. North Holland, 675–677.

Jünger, M. 1985. *Ployhedral Combinators and the Acyclic Subgraph Problem*. Heldermann, Berlin.

KARP, R. 1972. Reducibility among combinatorial problems. In *Procs. Complexity of Computer Computations*. 85–103.

KEMENY, J. 1959. Mathematics without numbers. *Daedalus 88*, 571–591.

KEMENY, J. AND SNELL, J. 1962. *Mathematical models in the social sciences*. Blaisdell, New York. Reprinted by MIT press, Cambridge, 1972.

KENYON-MATHIEU, C. AND SCHUDY, W. 2007. How to rank with few errors: a PTAS for weighted feedback arc set on tournaments. In *Procs. 39^{th} ACM STOC*. 95–103.

KORTE, B. 1979. Approximation algorithms for discrete optimization problems. *Ann. Discrete Math 4*, 85–120.

LEIGHTON, T. AND RAO, S. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM 46,* 6, 787–832.

LEMPEL, A. AND CEDERBAUM, I. 1966. Minimum feedback arc and vertex set of a directed graph. *IEEE Trans. Circuit Theory 13,* 4, 399–403.

MATHIEU, C. AND SCHUDY, W. 2008. Yet another algorithm for dense max cut: Go greedy. In *Procs. 19^{th} ACM-SIAM SODA*. 176–182.

NEWMAN, A. 2001. The maximum acyclic subgraph problem and degree-3 graphs. In *Procs. APPROX*. 147–158.

NEWMAN, A. 2004. Cuts and orderings: On semidefinite relaxations for the linear ordering problem. In *Procs. APPROX*. 195–206.

NEWMAN, A. AND VEMPALA, S. 2001. Fences are futile: on relaxations for the linear ordering problem. In *Procs. Integer Programming and Combinatorial Optimization (IPCO)*. 333–347.

REID, K. 1969. On sets of arcs containing no cycles in tournaments. *Canad. Math Bulletin 12*, 261–264.

REID, K. AND PARKER, E. 1970. Disproof of a conjecture of Erdös and Moser on tournaments. *J. Combin. Theory 9*, 225–238.

SESHU, S. AND REED, M. B. 1961. *Linear Graphs and Electrical Networks*. Addison-Wesley, Reading, MA.

SEYMOUR, P. D. 1995. Packing directed circuits fractionally. *Combinatorica 15*, 281–288.

SLATER, P. 1961. Inconsistencies in a schedule of paired comparisons. *Biometrika 48*, 303–312.

SPENCER, J. 1971. Optimal ranking of tournaments. *Networks 1*, 135–138.

SPENCER, J. 1980. Optimal ranking of unrankable tournaments. *Period. Math. Hungar. 11,* 2, 131–144.

SUGIYAMA, K., TAGAWA, S., AND TODA, M. 1981. Methods for visual understanding of hierachical system structures. *IEEE Trans. System Man Cybern. 11,* 2, 109–125.

VAN ZUYLEN, A., HEGDE, R., JAIN, K., AND WILLIAMSON, D. P. 2007. Deterministic pivoting algorithms for constrained ranking and clustering problems. In *Procs. ACM-SIAM SODA*. 405–414.

VAN ZUYLEN, A. AND WILLIAMSON, D. P. 2007. Deterministic algorithms for rank aggregation and other ranking and clustering problems. In *Procs. Workshop on Approximation and Online Algorithms*. 260–273.

YOUNG, P. 1995. Optimal voting rules. *The Journal of Economic Perspectives 9,* 1, 51–64.

YOUNGER, D. H. 1963. Minimum feedback arc sets for a directed graph. *IEEE Trans. Circuit Theory 10*, 238–245.