Electronic Colloquium on Computational Complexity, Report No. 144 (2006)

How to rank with few errors

A PTAS for Weighted Feedback Arc Set on Tournaments

Claire Kenyon-Mathieu
claire@cs.brown.edu

Warren Schudy
ws@cs.brown.edu

Computer Science
Brown University
Providence, RI 02912-1910

November 20, 2006

Abstract

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 NP-hard 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.

ISSN 1433-8092

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 NP-hard minimum feedback arc set (FAS) problem on tournaments.

For a general directed graph, the FAS problem consists of removing the fewest number of edges so as to make the graph acyclic, and comes up in applications such as scheduling [24] and graph layout [45, 16] (see also [37, 6, 30]). The problem has been much studied, both in the mathematical programming community [46, 26, 27, 30, 34] and the approximation algorithms community [36, 41, 22]. Indeed, the problem is NP-complete [31], and the reduction, from vertex cover, is approximation-preserving, so we know that the problem is also hard to approximate better than 1.36... [18].

The restriction of FAS to tournament inputs is an important, more tractable special case. A *tournament* is a directed graph where every pair of vertices is connected by exactly one of the two possible directed edges. The FAS problem on tournaments has a long and rich history, starting in the early 1960s in combinatorics [40, 47] and statistics [42]. In combinatorics and discrete probability, much early work focused on worst-case tournaments, starting with Erdös and Moon and culminating with work by Fernandez de la Vega [21, 38, 39, 29, 43, 44, 23]. 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 or animal's preferences for types of food. Early interesting heuristics due to Slater and Alway can be found in [42]. Unfortunately, even on tournaments, it was recently shown by Alon, building on work by Ailon, Charikar and Newman, that FAS is still NP-hard [1, 3] (see also [14]). The best approximation algorithms achieve constant factor approximations [1, 15, 48]: in the randomized setting, 2.5-approximation algorithms [1]; in the deterministic setting, 3-approximation algorithms [1, 48, 49]. Our main result is a polynomial time approximation scheme (PTAS) for this problem: thus the problem really is easier in tournaments than on general graphs.

Here is a generalization to weighted tournaments. (The unweighted problem is the special case where $w_{ij} = 0$ if $(i, j) \in E$ and $w_{ij} = 1$ 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.

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

We note that the previously known approximation algorithm results all extend to weighted tournaments for $b = 1$.

The main motivating 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 [19, 20], and desires to aggregate the input rankings into a single output ranking that is similar to all of the input rankings. This ancient problem was already studied in the context of voting by Borda

[9] and Condorcet [13] in the 18th century, and has been the object of renewed interest recently because of applications in learning [12]. By "similar to all of the input rankings", one means that the ranking should have minimum average distance from the input rankings, for some notion of distance. The possibly most natural notion of distance is the number of pairs of vertices that are in different orders: this leads to *Kemeny-Young rank aggregation* [32, 33].[1] For rank aggregation, the best known algorithms also achieve constant factor approximations: 4/3 for randomized algorithms [1, 2] and 2 for deterministic algorithms [1, 48, 49].

We improve on these results by designing a polynomial-time approximation scheme.

Theorem 1 (PTAS)**.** *There is a randomized polynomial-time approximation scheme for minimum weighted Feedback Arc Set on tournaments and for Kemeny-Young rank aggregation. Given $\epsilon > 0$, the algorithm outputs, in time $\tilde{O}((1/\epsilon)n^6 + 2^{2^{\tilde{O}(b/\epsilon)}} n^4)$, an ordering whose exptected cost is less than $(1+\epsilon)OPT$. The algorithm can be derandomized at the cost of increasing the running time by a factor of $n^{2^{\tilde{O}(1/\epsilon)}}$.*

Our algorithm uses existing constant factor approximation algorithms. In addition, it uses existing polynomial-time approximation schemes for the complementary maximization problem (a.k.a. max acyclic subgraph tournaments), due to Arora, Frieze and Kaplan [5] and to Frieze and Kannan [25]. Moreover, our central algorithm, Algorithm 2, also uses the divide-and-conquer paradigm. Of course, this has been previously used in this setting, in partitioning algorithms in general graphs [36] and in a Quicksort-type algorithm [1], but both of these construction are top-down, whereas our algorithm is bottom-up, which gives it a very different flavor. Finally, it also relies on a simple but useful technique, used already in 1961 by Slater and Alway [42]: 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 [28, 47, 19, 20].)

We can feel lucky that the FAS problem on tournaments turns out to be so easy as to have an approximation scheme: In contrast to Theorem 1, the related problem of feedback *vertex* set is hard to approximate even on tournaments (it has a 2.5-approximation algorithm [11]).

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

2 Algorithm

Our algorithm uses that additive approximation algorithms of [5, 25] as a subroutine; these algorithms are based on sampling. This extended abstract uses the derandomized additive error algorithm for simplicity. Appendix D discusses how to extend our analysis to use the randomized version, yielding the runtime in Theorem 1.

Theorem 2 (Small additive error)**.** *[5, 25] There is a randomized polynomial-time approximation scheme for maximum acyclic subgraph on tournaments. Given $\beta, \eta > 0$, the algorithm AddApprox outputs, in polynomial time ($n^2 + 2^{O(1/\beta^2)} \log 1/\eta$ in [25]), an ordering whose cost is, with probability at least $1-\eta$, less than $OPT + \beta n^2$. The algorithm can be derandomized, which replaces the $\log 1/\eta$ in the runtime with $n^{\tilde{O}(1/\epsilon^2)}$.*

[1] Kemeny-Young rank aggregation was invented as a voting system but has serious weaknesses as such: see [7, 8, 35] for the strategy issues that cause this.

Given: Fixed parameters $\epsilon > 0$ and $b \in (0, 1]$.
Input: A weighted tournament.
Round each weight to the nearest integer multiple of $\epsilon b/n^2$.
$\pi \leftarrow$ output of any constant factor approximation algorithm[2][1, 15]. ;
While there exists a cost-decreasing move, do that move. The two types of moves are:

1. **Single vertex moves.** Choose a vertex x and a rank j, take x out of the ordering π and insert it back in so that its rank is j.
2. **Additive approximation.** Choose two integers $i < j$; let U be the set of vertices whose current ranks are in $[i, j]$. Execute the derandomized version of algorithm AddApprox on U, with $\beta = 9^{-r_{\max}}\epsilon^3$. Let π'_U denote the result. Replace the restriction π_U of π to U by π'_U.

Output: π.

Algorithm 1: Polynomial time approximation scheme of Theorem 1 (PTAS)

Any total order R can equivalently be defined by an ordering $\pi : V \to \{1, \ldots, n\}$ such that $\pi(x)$ is the position of vertex x: $\pi(x) < \pi(y)$ iff $R(x, y)$.

Definition 3. 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 use a parameter $r_{\max} \equiv \log_{3/2}(1/\epsilon^2)/\epsilon = O(1/\epsilon \log(1/\epsilon))$. 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.) Our (somewhat faster) randomized PTAS is given in Algorithm 2. Curiously, if the constant factor approximation algorithm used is the sorting by in degree algorithms from [15], the initial order π^{local} computed in Algorithm 2 is precisely the order returned by the heuristic algorithm created by Slater and Alway [42] in 1961! For the analysis, it is enough to study Algorithm 2, since the result for Algorithm 1 follows as a simple corollary:

Claim 4. *If Algorithm 2 outputs an ordering whose cost is $\leq (1+\epsilon)OPT$ with positive probability, then Algorithm 1 outputs an ordering with cost $\leq (1+\epsilon)OPT$.*

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 DivideAndConquer 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 $\leq (1+\epsilon)$OPT. □

Actually, the recursive bottom-up nature of Algorithm 2 makes it quite difficult to analyze directly, and so we will instead analyze a different, top-down process, Algorithm 3. This is a virtual

[2]This algorithm actually still works if we start with an arbitrary π, not necessarily a constant factor approximation. The randomized algorithm, however, needs the constant-factor algorithm to work.

Given: Fixed parameter $\epsilon > 0$.
Input: A weighted tournament
Round each weight to the nearest integer multiple of $\epsilon b/n^2$.
$\pi \leftarrow$ output of any constant factor approximation algorithm [1, 15].
Apply profitable single vertex moves to π until a local optimum π^{local} is reached.
Output: **DivideAndConquer**(1,n)

DivideAndConquer(i, j) =
begin
if *i=j* **then return** i
For any ℓ, m, let $V_{\ell,m}$ denote the vertices x such that $\pi^{local}(x) \in [\ell, m]$
Let K be the values of k such that V_{ik} and $V_{k+1,j}$ both have size $\geq |V_{ij}|/3$.
Choose k uniformly at random from K.
$\rho_1 \leftarrow$ Output of derandomized AddApprox executed on V_{ij} with $\beta = 9^{-r_{\max}}\epsilon^3$
$\rho_2 \leftarrow$ **DivideAndConquer**(i, k) concatenated with **DivideAndConquer**(k+1, j)
return ρ_1 *or* ρ_2*, whichever has lower cost.*
end

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

Given: Fixed parameter $\epsilon > 0$.
Input: A weighted tournament; and π^*, an optimal ordering
Round each weight to the nearest integer multiple of $\epsilon b/n^2$.
$\pi \leftarrow$ output of any constant factor approximation algorithm [1, 15].
Apply profitable single vertex moves to π until a local optimum π^{local} is reached.
Let $\alpha = 9^{-r}$, where r is uniformly distributed in $\{0, 1, \ldots, r_{\max} = \log_{3/2}(1/\epsilon^2)/\epsilon\}$.
Output: π^{out} = **DivideAndConquer**(1,n)

DivideAndConquer(i, j) =
begin
if *i=j* **then return** i
For any ℓ, m, let $V_{\ell,m}$ denote the vertices such that $\pi^{local}(x) \in [\ell, m]$
Let K be the values of k such that V_{ik} and $V_{k+1,j}$ both have size $\geq |V_{ij}|/3$.
Choose k uniformly at random from K.
$\rho_1 \leftarrow$ Output of derandomized AddApprox executed on V_{ij} with $\beta = 9^{-r_{\max}}\epsilon^3$
$\rho_2 \leftarrow$ **DivideAndConquer**(i, k) concatenated with **DivideAndConquer**(k+1, j)
if $F_{V_{i,j}} \geq \alpha\epsilon^2|V_{i,j}|^2$ **then**
return ρ_1 (V_{ij} is called a leaf)
else
return ρ_2 (V_{ij} is called an internal node)
end
end

Algorithm 3: The virtual algorithm. Vertical lines in left margin denote differences from Algorithm 2.

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! The notation $F_{V_{i,j}}$, which depends on π^{local} and π^*, will be defined later, but we do not need to know what it means in order to reduce the analysis of Algorithm 2 to Algorithm 3.

Claim 5. *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 Lemma, proven in Section 3.1. 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.

Lemma 6. *For the orderings defined in Algorithm 3:*

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

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 $\mathbf{E}\left[C(\pi^{out})\right] \leq C(\pi^*)(1 + O(\epsilon))$, hence Algorithm 3 is an approximation scheme. By Claim 5, so is Algorithm 2, and by Claim 4, so is Algorithm 1.

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

3 Analysis of the virtual algorithm

We first deal with the rounding. (Proof deferred to Appendix A.)

Lemma 7 (Rounding)**.** *Let C denote the cost function with the original weights and $\tilde{C}$ 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. Assume that $\tilde{C}(\pi) \leq (1 + \epsilon)\tilde{OPT}$. Then $C(\pi) \leq (1 + 3\epsilon)OPT$.*

Thanks to Lemma 7, we can analyze the rounded weights as if they were the original weights. For the remainder of the paper we analyze the operation of the algorithm on the rounded graph as if it were the original graph. It remains to prove Lemma 6.

3.1 Proof of Lemma 6

For any subset S of vertices, let $F_S = \sum_{x \in S} |\pi^{local}(x) - \pi^*(x)|$ (this is often called *Spearman's footrule distance* between the two orderings). This completes the specification of the stopping condition used in Algorithm 3.

The footrule distance is related to the cost of the two orderings (proof in Appendix A):

Claim 8 (Tournaments)**.** *$F_V \leq (2/b)(C_V(\pi^*) + C_V(\pi^{local}))$.*

The following claim motivates the stopping condition of Algorithm 3.

Claim 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 $\alpha_{min} = 9^{-r_{\max}} \leq \alpha$:

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

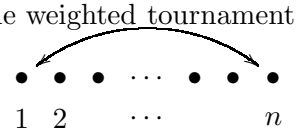
□

We are now ready for some intuition on 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's a constant factor approximation), and so, by Claim 8, the footrule distance will also be small, 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 divide steps are not too bad.

But, 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^*, and so 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^*, when k is chosen randomly. Thus, we expect that only a few vertices will be misplaced during the divide step.

How costly can it be, if, say, 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: then, 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.

More generally, define the *displacement graph* that has nodes set $[1, n]$, and has an arc from i to j whenever some vertex $x \in V$ has position i in π^{out} and position j in π^*. (The displacement graph should not be confused with the weighted tournament graph.) For example:

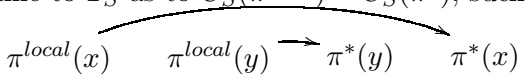


In this example, the first divide step will commit the vertex whose rank in π^* is n to have rank $\leq k$ in π^{out}, and commit the vertex whose rank in π^* is 1 to have rank $> k$ in π^{out}, so it may conceivably

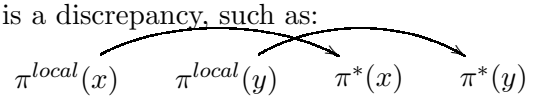
incur a cost of $\Theta(n)$. Again, local optimality with respect to single-vertex moves comes to the rescue and helps estimate the difference in cost between π^* and π^{out} by considering the displacement of one vertex at a time, noting the changing costs due to each individual displacement but ignoring the interactions between several vertices displacements. This is in a sense a first order approximation to the change in cost, and will be formally captured in a term of our analysis which we denote by T_V, defined as the *n*egation of the sum of the costs of moving each vertex of S according to the displacement graph from π^{local} to π^*. The negation is there because we are trying to upperbound $C(\pi^{local}) - C(\pi^*)$, but f is defined as part of the cost $C(\pi^*)$ relative to $C(\pi^{local})$.

What did we forget in this discussion? We forgot that when we move x between position 1 and position $\ell = \pi^*(x)$, the set of vertices y such that the order of $\{x, y\}$ changes is not exactly the vertices that are between 1 and ℓ in π^{local} because some vertices may have $\pi^{local}(y) < \ell$ but $\pi^*(y) > \ell$. This is what we call a "crossing", and we will need a corrective term, which we will denote by Φ_V, taking those into account.

More precisely, how does T_S compare to $C_S(\pi^{local}) - C_S(\pi^*)$? Both can be expanded as sums over pairs of vertices $\{x, y\}$ of S, so the comparison rests on which pairs are counted in each one. Some pairs 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:



Considering all of the possible permutations of $\{\pi^{local}(x), \pi^{local}(y), \pi^*(x), \pi^*(y)\}$ shows that the pairs for which there is a discrepancy correspond to a pair of arcs crossing in the displacement graph; the corresponding vertices are called a *crossing pair*. Formalizing this reasoning leads to Lemma 10 below (formal definition of "crossing pair" and proof in Appendix A.)

Let $f_\pi(e)$ denotes the cost, for edge $e = \{x, y\}$ of switching the relative order of x and y:

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

Lemma 10 (Mistake Decomposition)**.** *We have:*

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

where $\Phi_S = \sum_{x,y \in S \text{ and crossing pair}} -\delta_{xy} f_{\pi^{local}}(x, y)$*, and*

$$\delta_{xy} = \begin{cases} +1 & \text{if } \pi^{local}(x) < \pi^*(y) < \pi^*(x) < \pi^{local}(y) \\ -1 & \text{otherwise.} \end{cases}$$

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

$$\Phi_V - \mathbf{E}_B\left[\sum_{S\ leaf} \Phi_S\right] \leq 96\frac{\gamma}{1-\gamma}\epsilon F_V.$$

Lemma 12. *For a random execution defined by a random choice of (B, α), we have*

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

Now we can prove Lemma 6: Since Lemma 11 is true for every α, it is also true in expectation over α. Adding to Lemma 12 and substituting Lemma 10 to both sides yields:

$$C_V(\pi^{local}) - C_V(\pi^*) - \mathbf{E}_{B,\alpha}\left[\sum_{S\ \text{leaf}} (C_S(\pi^{local}) - C_S(\pi^*))\right] \leq (14 + 96\frac{\gamma}{1-\gamma})\epsilon F_V \tag{1}$$

Claim 13.

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

Proof Sketch. These orderings only differ inside the leaves. □

Now take the expectation of both sides of Equation (1) and apply Claim 13:

$$\mathbf{E}\left[C_V(\pi^{out})\right] - C_V(\pi^*) - \mathbf{E}_{B,\alpha}\left[\sum_{S\ \text{leaf}} (C_S(\pi^{out}) - C_S(\pi^*))\right] \leq (14 + 96\frac{\gamma}{1-\gamma})\epsilon F_V$$

By Claim 9, the expectation on the left-hand side is bounded above by $\mathbf{E}_{B,\alpha}\left[\sum_{S\ \text{leaf}} \epsilon F_S\right] = \epsilon\mathbf{E}_{B,\alpha}\left[F_V\right]$, which is independent of B, α so the expectation can now be dropped. Therefore

$$\mathbf{E}\left[C_V(\pi^{out})\right] - C_V(\pi^*) \leq (14 + 96\frac{\gamma}{1-\gamma} + 1)\epsilon F_V$$

Using Claim 8:

$$\mathbf{E}\left[C_V(\pi^{out})\right] - C_V(\pi^*) \leq (96\frac{\gamma}{1-\gamma} + 15)\epsilon\frac{2}{b}(C_V(\pi^{local}) + C_V(\pi^*)).$$

This concludes the proof of Lemma 6. It only remains to prove Lemmas 11 and 12.

3.2 Proof of Lemma 11 (bounding $\Phi_V - \sum \Phi_S$)

Our proof is by induction on the nodes of the recursion tree. We will need to bound the increase in second-order mistakes due to a single divide step. The number of vertices crossing a particular point in an ordering is important.

Notation 14. Let $\psi_S(k)$ (resp. $\psi_S^R(k)$)denote the number of vertices in S that are strictly after (resp. before) position k in the π^{local} ordering but before (resp. strictly after) position k in the π^* ordering: $\psi_S = |\{\, x \in S |\ (\pi^*(x) \le k < \pi^{local}(x)\,\}|$ and $\psi_S^R = |\{\, x \in S |\ (\pi^{local}(x) \le k < \pi^*(x)\,\}|$.

To prove the next lemma we need the following lower bound on the number of different possible choices for k.

Claim 15. *The set K from which Algorithm 3 randomly chooses k satisfies $|K| \ge |V_{ij}|/4$ whenever $i \ne j$.*

Proof. Asymptotically $|K| \approx |V_{ij}|/3$. Worst case for this claim is $|V_{ij}| = 4$ which leads to $|K| = 1$. Proof in Appendix B. □

Lemma 16 (Core)**.** *Let $S = \{x : \pi^{local}(x) \in [\ell, r]\}$, let k be a random element of K. Let $L = \{x \in S, \pi^{local}(x) \le k\}$ and $R = S \setminus L$. Then:*

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

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

Proof. For $U \in \{S, L, R\}$ recall that:

$$\Phi_U = \sum_{x,y \in U, x,y \text{ crossing pair}} -\delta_{xy} f_{\pi^{local}}(x, y)$$

Since $|\delta_{xy}| \le 1$ and $|f_{\pi^{local}}(x, y)| \le 1$, we can write

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

where we write $L = L_k$ and $R = R_k$ to make the dependency in k explicit. By Claim 15, we have:

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

In such a triple (x, y, k), we must have that k either belongs to $[\min(\pi^{local}(x), \pi^*(x)), \max(\pi^{local}(x), \pi^*(x)]$ or to $[\min(\pi^{local}(y), \pi^*(y)), \max(\pi^{local}(y), \pi^*(y)]$. 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 $\psi_S(\pi^{local}(x))$ or in $\psi_S^R(\pi^{local}(x))$ or in $\psi_S(\pi^*(x))$ or in $\psi_S^R(\pi^*(x))$. Taking the union of these cases yields the lemma. □

Lemma 17. *For any S and k, we have: $\psi_S(k) \le F_S^{1/2}$ and $\psi_S^R(k) \le F_S^{1/2}$.*

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 $\sqrt{F_S}$. See Appendix B for a full proof. □

Corollary 18.

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

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* $\alpha \in [\alpha_{min}, 1]$, $\Phi_V \leq \mathbf{E}_B \left[\sum_S \textit{ leaf}[96\frac{\gamma}{1-\gamma}\epsilon F_S + \Phi_S] \quad \textit{given } \alpha\right]$, *where* $\gamma = \sqrt{5}/3$.

Proof Sketch. The factor of $|S|$ in the denominator causes $F_S^{3/2}/|S|$ to increase exponentially as the vertices are subdivided, so by a geometric series argument the mistakes can be bounded by $O(1)$ time the sum of $F_S^{3/2}/|S|$ over the leaves. At a leaf S, $F_S^{3/2}/|S| \approx \epsilon F_S$ by the stopping condition. Full proof in Appendix B. □

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

3.3 Proof of Lemma 12 (bounding $T_V - \sum T_S$)

Recall that by local optimality of the single vertex moves $T_V \leq 0$. Unfortunately, we need to bound $T_V - \sum_{S \text{ leaf}} T_S$, which is not necessarily negative because $\sum_{S \text{ leaf}} T_S$ might be more negative than T_V is. We split $T_V - \sum_{S \text{ leaf}} T_S$ into separate terms for each vertex and bound each term by the displacements of that vertex times ϵ.

In this part, we will prove that $T_V - E_{B,\alpha}(\sum_{S \text{ leaf}} T_S) \leq 11\epsilon F_V$, which implies Lemma 12. First we see that by definition of T_V and $\{T_S\}$,

$$T_V - \mathbf{E}_{B,\alpha}\left[\sum_{S \text{ leaf}} T_S\right] = \mathbf{E}_{B,\alpha}\left[\sum_y \sum_{\substack{x : \pi^{local}(x) \text{ between } \pi^{local}(y) \text{ and } \pi^*(y) \\ x \notin \text{ leaf containing } y}} f_{\pi^{local}}(x, y)\right].$$

Let $s(y)$ be the size of the leaf containing y. 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 big leaves ($\epsilon s(y) > |\pi^*(y) - \pi^{local}(y)|$), small leaves ($|\pi^*(y) - \pi^{local}(y)| > s(y)/\epsilon$) and intermediate leaves ($\epsilon s(y) \leq |\pi^*(y) - \pi^{local}(y)| \leq s(y)/\epsilon$). We handle the small leaves by showing that small leaves are essentially negligible so the cost of moving y from where y leaves its leaf to $\pi^*(y)$ is approximately its cost of moving from $\pi^{local}(y)$ to $\pi^*(y)$, which is non-positive by local optimality. We handle the big leaves by proving that the probability of $\pi^{local}(y)$ and $\pi^*(y)$ being in different leaves (and hence able to contribute to the sum) is $O(\epsilon)$. We bound the contribution of the intermediate leaves by using the variation in α to force a variation in the leaf size, making this event unlikely. See Appendix C for details.

Acknowledgements

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

References

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

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

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

[4] N. Alon, J. Spencer. The probabilistic method, second edition. Wiley-Interscience. 2000.

[5] 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.

[6] 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.

[7] John J. Bartholdi, J. A. Orlin. Single transferable vote resists strategic voting. Social Choice and Welfare 8:341-354 (1991).

[8] JJ Bartholdi, CA Tovey, MA Trick. The computational difficulty of manipulating an election. Social Choice and Welfare, 6, 3, 1989.

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

[10] J.C. Bermond, Ordres à distance minimum d'un tournoi et graphes partiels sans circuits maximaux, Math. Sci. Humaines 37 (1972), 5-25.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

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

[22] 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.

[23] 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).

[24] 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.

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

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

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

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

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

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

[31] 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.

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

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

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

[35] Lars-Gunnar Svensson. The Proof of the Gibbard-Satterthwaite Theorem Revisited. Working Paper No 1999:1, Department of Economics, Lund University. 1999. http://swopec.hhs.se/lunewp/papers/lunewp1999_001.pdf

[36] 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.

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

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

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

[40] 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.

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

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

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

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

[45] 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.

[46] A.W.Tucker. On directed graphs and integer programs. Presented at 1960 Symposium on Combinatorial Problems, Princeton University, cited in [47].

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

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

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

A Proofs and calculations - Section 3.1

Proof of Lemma 7

Proof. By definition of rounding, for every ρ we have:

$$|\tilde{C}(\rho) - C(\rho)| \leq \binom{n}{2} \epsilon b/(2n^2) \leq \epsilon b/4. \tag{2}$$

By definition of $O\tilde{P}T$, for every ρ we have $O\tilde{P}T \leq \tilde{C}(\rho) \leq C(\rho) + \epsilon b/4$. Minimizing over ρ gives

$$O\tilde{P}T \leq OPT + \epsilon b/4. \tag{3}$$

We first analyze the case where $OPT \geq b(1/2 - \epsilon)$. Using Equation 2, our assumption on the output ordering π, and Equation 3, we get

$$C(\pi) \leq \tilde{C}(\pi) + \epsilon b/4 \leq (1+\epsilon)O\tilde{P}T + \epsilon b/4 \leq (1+\epsilon)OPT + (1+\epsilon)\epsilon b/4 + \epsilon b/4.$$

Since $OPT \geq b(1/2 - \epsilon)$, this implies

$$C(\pi) \leq OPT\left[1 + \epsilon(1 + \frac{1+\epsilon/2}{1-2\epsilon})\right] \leq OPT(1+3\epsilon).$$

We next analyze the case where $OPT < b(1/2 - \epsilon)$. Let π^* be the cheapest ordering, such that $C(\pi^*) = OPT$. By definition of b, for every pair $\{x, y\}$ we have $\max(w_{xy}, w_{yx}) \geq b/2 > OPT$, so π^* must be exactly the ordering such that $\pi^*(x) < \pi^*(y)$ iff $w_{xy} < w_{yx}$, and has cost exactly $\sum_{x,y} \min(w_{xy}, w_{yx})$, and for every pair we must have $\min(x, y) < b(1/2 - \epsilon)$. Consider any other ordering ρ: for every pair, ρ costs at least as much as π^*, and since $\rho \neq \pi^*$, there must exist at least one pair $\{x, y\}$ where ρ pays $\max(w_{xy}, w_{yx}) > \min(w_{xy}, w_{yx}) + \epsilon b$, and so $C(\rho) > C(\pi^*) + \epsilon b$. From Equation 2, it follows that $\tilde{C}(\rho) > \tilde{C}(\pi^*) + \epsilon b/2$. Since by Equation 3 we have $O\tilde{P}T = \tilde{C}(\pi^*) < b/2$, this implies that $\tilde{C}(\rho) > (1+\epsilon)O\tilde{P}T$. But by assumption, the algorithm outputs a ordering π such that $\tilde{C}(\pi) \leq (1+\epsilon)O\tilde{P}T$. So it has no choice but to output $\pi = \pi^*$, the optimal ordering: the algorithm is optimal in this case. □

Proof of Claim 8

Proof. By [17] Theorem 2 relating Spearman's footrule and Kendall-Tau, we have:

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

where the second inequality follows from the fact that $\frac{1}{b}(w_{ij} + w_{ji}) \geq 1$.

□

Proof of Lemma 10

We need a precise definition for Φ that formalizes the notion of crossing pairs:

$$\Phi_S = \sum_{x,y \in S : \pi^{local}(x) < \pi^{local}(y)} -f_{\pi^{local}}(x,y) \cdot \cdot \begin{cases} 1 & \pi^{local}(x) < \pi^*(y) < \pi^*(x) < \pi^{local}(y) \\ 0 & \text{An even number of } \{\pi^*(x) > \pi^*(y), \pi^*(y) \le \pi^{local}(x), \pi^{local}(y) \le \pi^*(x)\} \text{ are true} \\ -1 & \text{otherwise} \end{cases}$$

Proof. First note that $\Delta C \equiv C_S(\pi^{local}) - C_S(\pi^*) = \sum_{x,y \in S : \pi^{local}(x) < \pi^{local}(y) \text{ and } \pi^*(x) > \pi^*(y)} -f_{\pi^{local}}(x,y)$. By definition, we have $T_S = \sum_{x \in S} \sum_{y \in S : \pi^{local}(y) \text{ between } \pi^{local}(x) \text{ and } \pi^*(x)} -f_{\pi^{local}}(x,y)$. To show equality, it suffices to show that each pair (x,y) appears the same number of times in the sum for ΔC that it does in the sums for the T_S and Φ_S. First rewrite T_S by splitting the sum into two parts depending on $\pi^{local}(x)$ is before or after $\pi^*(x)$, then swap the names x and y in one of the sums to yield $T_S = T_1 + T_2$ where $T_1 = \sum_{x,y \in S : \pi^{local}(x) < \pi^{local}(y) \le \pi^*(x)} -f_{\pi^{local}}(x,y)$ and $T_2 = \sum_{x,y \in S : \pi^*(y) \le \pi^{local}(x) < \pi^{local}(y)} -f_{\pi^{local}}(x,y)$. Note that every pair (x,y) in all three sums has $\pi^{local}(x) < \pi^{local}(y)$. We divide the pairs into eight cases based on the truth of the inequalities $\pi^*(x) > \pi^*(y)$, $\pi^*(x) \ge \pi^{local}(y)$, and $\pi^*(y) \le \pi^{local}(x)$. The eight cases are shown in the following table. T indicates that the inequality in the heading is true, integers indicate the number of times pairs that belong in that row occur in the sum in that column, and dots indicate false or zero.

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

Note that the fourth row is defined by three inequalities that combine with $\pi^{local}(x) < \pi^{local}(y)$ to form a contradiction. This explains why the fourth row need not add up.

□

B Proofs and calculations - Section 3.2 (bounding $\Phi_V - \sum \Phi_S$)

Proof of Claim 15 ($K \ge |V_{ij}|/4$)

Proof. Let $n = |V_{ij}|$. There are $2\lceil n/3 \rceil$ vertices that are forced to be in one child or the other, so $|K| = n - 2\lceil n/3 \rceil + 1 = \begin{cases} \frac{n+3}{3} & n \bmod 3 = 0 \\ \frac{n-1}{3} & n \bmod 3 = 1 \\ \frac{n+1}{3} & n \bmod 3 = 2 \end{cases}$. The nontrivial case is $n \bmod 3 = 1$. The smallest $n \ge 2$ of this form is $n = 4$, so $|K|/n \ge 1/3 - 1/(3n) \ge 1/4$. □

Proof of Claim 17 ($\psi \leq \sqrt{F}$)

ψ_S^R is the same as ψ_S but with the roles of π^{local} and π^* reversed, so we need only prove the claim for ψ_S.

The following intermediate claim is useful:

Claim 20. $\sum_{0 \leq i \leq |V|} \psi_S(i) \leq F_S$.

Proof.

$$\psi_S(i) = \sum_{j \in S} \mathbb{1}\left(\pi^{local}(j) > i \text{ and } \pi^*(j) \leq i\right)$$

$$\sum_i \psi_S(i) = \sum_{j \in S} \sum_i \mathbb{1}\left(\pi^*(j) \leq i < \pi^{local}(j)\right) = \sum_{j \in S : \pi^{local}(j) > \pi^*(j)} (\pi^{local}(j) - \pi^*(j)) \leq \sum_{j \in S} |\pi^{local}(j) - \pi^*(j)| = F_S$$

□

A second helper claim:

Claim 21. $\forall i, |\psi_S(i) - \psi_S(i+1)| \leq 1$.

Proof.

$$\begin{aligned}
\psi_S(i) &= |\{j \in S | \pi^*(j) \leq i < \pi^{local}(j)\}| \\
&= |\{j \in S | \pi^{local}(j) > i+1 \text{ and } \pi^*(j) \leq i\}| + \mathbb{1}\left(\exists j \in S : \pi^{local}(j) = i+1 \text{ and } \pi^*(j) \leq i\right) \\
\psi_S(i+1) &= |\{j \in S | \pi^{local}(j) > i+1 \text{ and } \pi^*(j) \leq i+1\}| \\
&= |\{j \in S | \pi^{local}(j) > i+1 \text{ and } \pi^*(j) \leq i\}| \\
&\quad + \mathbb{1}\left(\exists j \in S : \pi^{local}(j) > i+1 \text{ and } \pi^*(j) = i+1\right) \\
\psi_S(i+1) - \psi_S(i) &= \mathbb{1}\left(\exists j \in S : \pi^{local}(j) > i+1 \text{ and } \pi^*(j) = i+1\right) \\
&\quad - \mathbb{1}\left(\exists j \in S : \pi^{local}(j) = i+1 \text{ and } \pi^*(j) \leq i\right)
\end{aligned}$$

Both terms are either 0 or 1, so the difference is in $[-1, 1]$.

□

Now proof of Claim 17:

Proof.

$\psi_S(n) = |\{j \in S | \pi^{local}(j) > n \text{ and } \pi^*(j) \leq n\}| = 0$ and $\psi_S(0) = |\{j \in S | \pi^{local}(j) > 0 \text{ and } \pi^*(j) \leq 0\}| = 0$

Suppose, for a contradiction, that there exists k such that $\psi_S(k) > \sqrt{F_S}$. Using Claim 21,

$$\sum \psi_S(i) \geq \psi_S(k) + 2 \sum_{i=1}^{\psi_S(k)-1} i = \frac{\psi_S(k)(\psi_S(k)+1) + (\psi_S(k)-1)\psi_S(k)}{2} = \psi_S(k)^2 > F_S.$$

This contradicts Claim 20.

□

Proof of Lemma 19

Claim 22 (Algebra)**.** *Let $\gamma = \sqrt{5}/3$. If $0 \le F_1 \le F$, $s > 0$, and $s_1 > 0$ such that $s_1, (s - s_1) \ge s/3$, then:*

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

Proof. Let $f(F_1, s_1) = F_1^{3/2}/s_1 + (F - F_1)^{3/2}/(s - s_1)$. Differentiating yields a unique stationary point of $F_1 = F/2$, $s_1 = s/2$, with value $\sqrt{2}F^{3/2}/s$. The minimum point along the boundary occurs when $s_1 = s/3$ and $F_1 = F/5$ with value $\sqrt{9/5}$. This is smaller than the stationary point in the interior, so this is the global minimum over this set. Therefore $\sqrt{9/5}F^{3/2}/s \le f(F_1, s_1)$, so $F^{3/2}/s \le \sqrt{5/9}\left[F_1^{3/2}/s_1 + (F - F_1)^{3/2}/(s - s_1)\right]$. □

Now the proof of Lemma 19:

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 $l \le m$, with $U = \{\, x |\ \ell \le \pi^{local}(x) \le m \,\}$ and $\mathbf{E}\left[\cdot | U \in B\right]$ meaning expectation conditioned on U being a leaf or internal node in execution tree B:

$$\Phi_U \le \mathbf{E}_B \left[\sum_{s \text{ internal descendant of } U} 32\frac{F_S^{3/2}}{|S|} + \sum_{s \text{ leaf descendant of } U} \Phi_S \middle| U \in B \right]. \qquad (4)$$

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, Φ_U is deterministic. Therefore:

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

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

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

Take expectation over k and rewriting:

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

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

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 4 with $\ell, m = 1, n$ and $U = V$ is bounded by $O(1)\epsilon F_V$. Therefore the expectation must also be bounded. For every internal node

S, by Claim 22 we have: $F_S^{3/2}/|S| \leq \gamma(F_{S_1}^{3/2}/|S_1| + F_{S_2}^{3/2}/|S_2|)$. Summing over the recursion tree, we obtain:

$$\sum_{S \text{ internal}} \frac{F_S^{3/2}}{|S|} \leq \sum_{S \text{ leaf}} \frac{F_S^{3/2}}{|S|}(\gamma + \gamma^2 + \cdots) \leq \frac{\gamma}{1-\gamma} \sum_{S \text{ leaf}} \frac{F_S^{3/2}}{|S|}.$$

If S is a leaf, then let P be the parent of S.[3] 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. □

C Proofs and calculations - Section 3.3 (bounding $T_V - \sum T_S$)

Lemma 23 (Small Leaves)**.** *For any execution (determined by B and α), we have*

$$\sum_{y:|\pi^*(y)-\pi^{local}(y)|>s(y)/\epsilon} \quad \sum_{\substack{x:\pi^{local}(x) \text{ between } \pi^{local}(y) \text{ and } \pi^*(y) \\ x \notin \text{ leaf containing } y}} -f_{\pi^{local}}(x,y) \leq \epsilon F_V. \qquad (5)$$

Proof. No single vertex move can improve π^{local}, so $\sum_{x:\pi^{local}(x) \text{ between } \pi^{local}(y) \text{ and } \pi^*(y)} -f_{\pi^{local}}(x,y) \leq 0$. Thus the left hand side of Equation 5 can be bounded by

$$\sum_{y:|\pi^*(y)-\pi^{out}(y)|>s(y)/\epsilon} \quad \sum_{\substack{x:\pi^{local}(x) \text{ between } \pi^{local}(y) \text{ and } \pi^*(y) \\ x \in \text{ leaf containing } y}} f_{\pi^{local}}(x,y) \leq \sum_{y:|\pi^*(y)-\pi^{out}(y)|>s(y)/\epsilon} s(y).$$

This is at most $\sum_y \epsilon|\pi^*(y) - \pi^{local}(y)| = \epsilon F_V$. □

Lemma 24 (Big Leaves)**.** *For any α and for a random B, we have:*

$$\mathbf{E}_B \left[\sum_{y:|\pi^*(y)-\pi^{local}(y)|<\epsilon s(y)} \quad \sum_{\substack{x:\pi^{local}(x) \text{ between } \pi^{local}(y) \text{ and } \pi^*(y) \\ x \notin \text{ leaf}(y)}} -f_{\pi^{local}}(x,y) \quad \text{given} \quad \alpha \right] \leq 12\epsilon F_V.$$

Proof. Let α be fixed. If $\pi^{local}(x)$ is between $\pi^{local}(y)$ and $\pi^*(y)$, but x is not in leaf(y), then ${\pi^{local}}^{-1}(\pi^*(y))$ 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

$$\sum_y |\pi^*(y) - \pi^{local}(y)| \cdot \mathbb{1}\left(|\pi^*(y) - \pi^{local}(y)| < \epsilon \cdot s(y) \text{ and } {\pi^{local}}^{-1}(\pi^*(y)) \notin \text{ leaf}(y)\right).$$

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

We will now argue that for any vertex y, the event E_1 that "$|\pi^*(y) - \pi^{local}(y)| < s(y)\epsilon$ and $\pi^{local^{-1}}(\pi^*(y)) \notin \text{leaf}(y)$" 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 $\pi^*(y)$.

Indeed, fix a vertex y and let $\ell = |\pi^*(y) - \pi^{local}(y)|$. Down the branch leading to the leaf of y, the algorithm uses a certain sequence of random variables $k_0, k_1, \ldots$. Instead of stopping when the stopping condition is reached[4], we conceptually extend the construction until the first time Z that the associated interval has size less than ℓ/ϵ. This defines a sequence of nested intervals $I_0 = [1, n]$, I_1, ..., I_Z.

The probability of event E_1 is bounded by the probability that the last interval does not contain $\pi^*(y)$, hence:

$$\Pr(E_1) \leq \sum_{i<Z} \Pr(\pi^*(y) \notin I_{i+1} | \pi^*(y) \in I_i).$$

The probability in the right hand side is that the random cutting point used by the construction falls between $\pi^{local}(y)$ and $\pi^*(y)$. There are only ℓ possibilities, so the probability is at most $4\ell/|I_i|$ by Claim 15. For $i = Z - 1$, by definition of Z we have $|I_i| > \ell/\epsilon$. For $i \leq Z - 1$, we can write

$$|I_i| = \frac{|I_i|}{|I_{i+1}|}\frac{|I_{i+1}|}{|I_{i+2}|}\cdots\frac{|I_{Z-2}|}{|I_{Z-1}|}|I_{Z-1}| \geq (3/2)^{Z-i-1}(\ell/\epsilon).$$

So

$$\Pr(E_1) \leq \sum_{i=0}^{Z-1} \frac{3\ell}{(3/2)^{Z-i-1}(\ell/\epsilon)} \leq 12\epsilon.$$

Summing yields the lemma.

□

Lemma 25 (Intermediate Leaves)**.** *For any set B and for a random α, we have*

$$\mathbf{E}_\alpha \left[\sum_{y:\epsilon s(y) \leq |\pi^*(y) - \pi^{local}(y)| \leq s(y)/\epsilon} \; \sum_{\substack{x\,:\,\pi^{local}(x) \text{ between } \pi^{local}(y) \text{ and } \pi^*(y) \\ x \notin \text{ leaf}(y)}} -f_{\pi^{local}}(x, y)) \right] \leq \epsilon F_V.$$

Proof. Let $B = (k_S)$ be fixed.

As in the beginning of the proof of Lemma 24, we can bound the expression on the left hand side by the expectation (over the random choice of α) of

$$\sum_y \begin{cases} 0 & \text{if } s(y) = n \\ \sum_{x:\pi^{local}(x) \text{ between } \pi^{local}(y) \text{ and } \pi^*(y)} -f_{\pi^{local}}(x, y) & \text{if } s(y) = 1 \\ |\pi^*(y) - \pi^{local}(y)| \cdot \mathbb{1}\left(\epsilon \cdot s(y) < |\pi^*(y) - \pi^{local}(y)| < (1/\epsilon) \cdot s(y)\right) \cdot & \\ \quad \mathbb{1}\left(\pi^{local^{-1}}(\pi^*(y)) \notin \text{ leaf}(y)\right) & \text{otherwise.} \end{cases}$$

[4]That would introduce some conditioning.

We will again argue that for any y, the event E_2 that "$\epsilon \cdot s(y) < |\pi^*(y) - \pi^{local}(y)| < (1/\epsilon) \cdot s(y)$ and $\pi^{local^{-1}}(\pi^*(y)) \notin$ 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.

The intuition is that the variation in α forces a variation in the leaf size, making this event unlikely.

Recall that $\alpha = 9^{-r}$, with r chosen uniformly at random in $\{0, 1, \ldots, \log_{3/2}(1/\epsilon^2)/\epsilon\}$.

Fix a vertex y and let $\ell = |\pi^*(y) - \pi^{local}(y)|$. Go down the branch leading to y. Let $S_0, S_1, S_2, \ldots$ be the sequence of nodes on that branch, and let $a_i = F_{S_i}/|S_i|^2$. Every time you go down from one node S_i to a child S_{i+1}, since $F_{S_{i+1}} \leq F_{S_i}$ and $|S_{i+1}| \geq (1/3)|S_i|$, you have $a_{i+1} \leq 9a_i$. By definition of the recursion tree, the leaf of y is the *first* node S_t along this branch where $a_t \geq \alpha\epsilon^2$ or which has size 1. Thus $a_t \leq 9a_{t-1} < 9\alpha\epsilon^2$, and so, for any $\alpha' \geq 9\alpha$, t is an internal node (if it has size greater than 1). Therefore a node t of size greater than 1 can be a leaf for only one value of $\alpha = 9^{-r}$.

Consider the node S of size 1. Even if $\ell < 1/\epsilon$, then whenever α is such that S is the leaf of y, by local optimality with respect to single vertex moves, y contributes ≤ 0 to the sum.

Consider the nodes S of size strictly between 1 and n, and such that $\epsilon\ell < |S| < \ell/\epsilon$: there are at most $\log_{3/2}(1/\epsilon^2)$ such nodes, hence event E_2 has probability at most $\log_{3/2}(1/\epsilon^2)/(\log_{3/2}(1/\epsilon^2)/\epsilon) = \epsilon$.

Summing concludes the proof.

□

D 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 AddApprox algorithm set $\eta = \delta/n$ for Algorithm 2 and $\eta = \delta\epsilon b/n^4$ for Algorithm 1. Consider the event E_0 stating that "During execution of the algorithm, every call to algorithm AddApprox yields a result within the stated bounds." Each call fails with probability at η. As shown in Section E, there are at most n (resp. $n^4 b\epsilon$) 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 conditional on E_0.

Using the randomized AddApprox algorithm introduces a complication because the proof of Claim 5 implicitly assumes determinism. This is easily remedied by reusing the random numbers used the previous time AddApprox was called on the same vertices.

E 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 $b\epsilon/n^2$, so the cost decreases by at least $b\epsilon/n^2$ at each iteration. Since the cost is always between n^2 and zero, the total number of iterations is bounded by $n^4/(b\epsilon)$.

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 $n^4/b\epsilon$ for the deterministic one (Algorithm 1). Thus the runtime is $O(n^3 f(n, (1/\epsilon)^{O(1/\epsilon)}, \delta/n^4))$ randomized and $O(n^6/(b\epsilon) f(n, (1/\epsilon)^{O(1/\epsilon)}, 0))$ 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 $n^2 - \log \eta 2^{O(1/\beta^2)}$ randomized in [25], or $n^2 + n^{\tilde{O}(1/\beta^2)} 2^{O(1/\beta^2)}$ for deterministic ($\eta = 0$).

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/