
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 30, no. 1, pp. 133–150 (2026)
DOI: 10.7155/jgaa.v30i1.3028

Improved Combinatorial Approximation for Feedback Arc Set and Rank Aggregation

Mojtaba Ostovari[1] *Alireza Zarei*[1]

[1]Department of Mathematical Sciences, Sharif University of Technology

Submitted: April 2025 Accepted: March 2026 Published: April 2026

Article type: Regular paper Communicated by: S. Cornelsen

Abstract. This paper introduces a simple and fast method to combinatorially approximate the feedback arc set problem on weighted tournaments, a well-known classical NP-hard problem. Our results are as follows:

- A combinatorial 3-approximation algorithm for the *minimum feedback arc set* problem with the expected running time of $O(|V|\log|V|)$ on tournaments that satisfy the probability constraints, where V is the set of vertices of the tournament. The previously best-known approximation factor was 4.
- A combinatorial 5/3-approximation algorithm for the *minimum feedback arc set* problem with the expected running time of $O(|V|\log|V|)$ on tournaments that satisfy both the probability and the triangle inequality constraints. The previously best-known approximation factor was 2.

Although these problems currently have a PTAS algorithm that returns a $1+\epsilon$ approximation solution, the running time is doubly exponential with respect to $1/\epsilon$, which is merely significant in theory. Moreover, there are LP-based algorithms that solve these problems with better approximation factors than ours, but at the expense of solving an LP with $O(|V|^2)$ variables and $O(|V|^3)$ constraints, which requires an excessive expenditure of time, making them inappropriate for practical use. There exist combinatorial algorithms with dramatically faster running times and larger approximation factors that are useful for practical use. The running time of our combinatorial algorithms is the same as that of the best-known combinatorial algorithms for this problem, but our algorithms have better approximation factors.

We have also shown that the proposed approach can be applied to other problems. Here, we use the method to approximate the rank aggregation problem and show the superiority of the result compared to current solutions.

E-mail addresses: m.ostovari@sharif.edu (Mojtaba Ostovari) zarei@sharif.edu (Alireza Zarei)

1 Introduction

A tournament is a complete weighted directed graph with exactly two directed edges (i, j) and (j, i) between any pair of distinct vertices i and j. A tournament can be represented by (V, w) where V is the set of its vertices, and weight function w indicates the weight of each edge, i.e. $w_{ij} \geq 0$ is the weight of edge (i, j). A simple tournament is a tournament such that for any pair of distinct vertices i and j, the weight of one of the edges (i, j) or (j, i) is 1, and the weight of the other one is 0. A tournament satisfies the probability constraints if $w_{ij} + w_{ji} = 1$ for any pair of distinct $i, j \in V$. A tournament satisfies the triangle inequality constraints if $w_{ik} \leq w_{ij} + w_{jk}$ for any triple of distinct $i, j, k \in V$.

A ranking on a finite set S of size n is a bijective function $\pi : S \to \{1, \ldots, n\}$ where $\pi(i)$ is the position of $i \in S$ in the ranking. An edge (i, j) in a tournament (V, w) is backward according to a ranking π on V, if $\pi(i) > \pi(j)$. The *cost* of a ranking π is defined as,

$$cost(\pi) = \sum_{\substack{i,j \in V \\ \pi(i) < \pi(j)}} w_{ji}.$$

A *feedback arc set* for a tournament is a set of edges whose removal makes the tournament acyclic. The weight of a *feedback arc set* is the sum of the weights of its edges. A *minimum feedback arc set* (FAST) is a *feedback arc set* with minimum possible weight. A *minimum feedback arc set* for (V, w) can also be described as a ranking π_{FAST} of V, whose *cost* is minimum among all rankings of V. Hence,

$$\pi_{FAST} = \operatorname*{argmin}_{\pi} cost(\pi).$$

The *minimum feedback arc set* problem on tournaments is a classical well-studied problem that has a rich history [11, 15, 18, 19], with numerous applications in ranking, preference aggregation, and decision-making [12, 10, 16, 8, 18]. This problem is a foundational problem due to its direct connection to real-world ranking systems where inconsistencies naturally arise from noisy or uncertain pairwise data. In many practical settings, the pairwise comparisons between elements are not deterministic but probabilistic. When the weight of each directed edge between vertices i and j in the tournament represents the probability that element i is preferred to element j. So, $w_{ij} + w_{ji} = 1$, we obtain tournaments that satisfy probability constraints. Such probabilistic tournaments model real phenomena in which preferences or outcomes are inherently uncertain. This framework arises in numerous contexts, including social choice and voting, sports ranking, psychometrics, and online recommendation systems, where outcomes are best expressed as likelihoods rather than absolute wins or losses. In these scenarios, the minimum feedback arc set problem finds an ordering that minimizes the number of upsets, effectively providing a maximum-likelihood ranking of the observed data [5].

These constrained tournament instances also have a particular application in information systems, where large-scale data processing often requires robust ranking mechanisms to handle inconsistent or noisy inputs. In rank aggregation, for example, multiple conflicting rankings from different sources, such as user preferences or algorithmic outputs are combined into a consensus ordering that minimizes the total Kendall tau distance. Here, the weights satisfy both probability and triangle inequality constraints. This directly applies to meta-search engines in information retrieval, where results from various search providers are merged to produce a unified list, improving query relevance and user satisfaction by minimizing discrepancies [2].

1.1 Previous Works

It has been shown that the *minimum feedback arc set* problem, even on simple tournaments, is NP-hard [2, 3]. Kenyon-Mathieu and Schudy [13] gave a polynomial-time approximation scheme (PTAS) for the *minimum feedback arc set* problem on tournaments which returns a $1+\epsilon$ approximation solution for any desired $\epsilon > 0$. However, the running time of this algorithm is doubly exponential in $1/\epsilon$.

Ailon *et al.* [2] gave an LP-based 2.5-approximation algorithm for the *minimum feedback arc set* problem on tournaments that satisfy the probability constraints. Recently, this was improved to 2.127 [14]. Also, there is an LP-based 1.5-approximation algorithm for the tournaments that satisfy the triangle inequality constraints [1]. These algorithms, in comparison to the PTAS algorithm, are much faster, although they still need to solve the LP formulation shown in relation (1), which is a linear programming for *minimum feedback arc set* problem on tournaments. Note that this LP has $O(|V|^2)$ variables and $O(|V|^3)$ constraints.

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i<j} x_{ij}w_{ji} + x_{ji}w_{ij} & \\ \text{subject to} & x_{ij} + x_{jk} + x_{ki} \geq 1 & \text{for all distinct } i,j,k \in V \\ & x_{ij} + x_{ji} = 1 & \text{for all distinct } i,j \in V \\ & 0 \leq x_{ij} & \text{for all distinct } i,j \in V \end{array} \tag{1}$$

Due to the high running time of the LP-based algorithms, more efficient algorithms have been proposed. Ailon *et al.* [2] introduced the first combinatorial constant factor approximation algorithm for the *feedback arc set* problem on tournaments, called KwikSort. It is a 5-approximation algorithm for tournaments that satisfy the probability constraints. For simple tournaments, it is a 3-approximation algorithm. It is also a 2-approximation algorithm if the tournaments satisfying both the probability and the triangle inequality constraints. This algorithm recursively finds a ranking which is an approximate solution for the *FAST* problem. It chooses a vertex $p \in V$ uniformly at random as a pivot, and, for any $u \in V - \{p\}$ if $w_{up} > w_{pu}$ it puts u to the left side of p, if $w_{up} < w_{pu}$ it puts u to the right side of p, and if $w_{up} = w_{pu}$ it puts u to the left side or the right side of p, arbitrarily. Then, it recursively finds the position of the vertices which are on the left side and on the right side of p, independently. Clearly, the expected running time of this algorithm is $O(|V| \log |V|)$.

Coppersmith *et al.* [7] gave another 5-approximation combinatorial algorithm for the *minimum feedback arc set* problem on tournaments having probability constraints. This algorithm simply sorts the vertices by their weighted in-degrees. The running time of this algorithm is $O(|V|^2)$ because we need $O(|V|)$ time to compute the in-degree of each vertex. Zuylen *et al.* [20] presented a 4-approximation combinatorial algorithm for the *minimum feedback arc set* problem on tournaments satisfying probability constraints. To the best of our knowledge, this algorithm was the best-known combinatorial algorithm for this problem. Table 1 summarizes these results along with the results from this paper.

1.2 Our Results

Our algorithm for the *FAST* problem, called QuickFAST, is similar to KwikSort, but we use different separating rules to assign vertices to the right or left sides of the pivot vertex. Our proposed separating rules improve the approximation factors for the tournaments satisfying the probability constraints from 4 to 3, and for the tournaments satisfying both the probability and triangle inequality constraints from 2 to 5/3.

Problem	**LP-Based Result**	**Previous Combinatorial Result**	**This Paper Combinatrial Result**
FAST (Probability Constraints)	2.5 [2], 2.127 [14]	5 [2, 7], 4 [20]	3
FAST (Probability and Triangle Inequality Constraints)	2 [3], 1.5 [1]	2 [2]	5/3

Table 1: Approximation factors of LP-based and combinatorial algorithms.

Designing proper probabilistic function is generally a common task in decision making in randomized algorithms, and mainly affects the efficiency and quality of the obtained result. Our method in this paper is similar to the method that Ailon *et al.* [2] used for their algorithm. We have adjusted and modified this method to be used for the analysis of our algorithm.

As the second part of this paper, we have applied the method for the rank aggregation problem which is in close connection with the FAST problem. The rank aggregation problem can be transformed to an instance of the FAST problem. Our algorithm for this problem, called QuickRank, uses QuickFAST as a subroutine and we show that the result is slightly better than the currently proposed best combinatorial algorithm [2].

Expanding on the landscape of algorithms for the minimum feedback arc set on the general weighted graph, several additional recent contributions offer complementary approaches to heuristic and scalable solutions. For example, Simpson et al. introduce an efficient depth-first search-based method for computing feedback arc sets in massive web-scale graphs, leveraging parallel processing to handle billions of nodes and edges, which is particularly effective for real-time applications in social networks and recommendation systems [17]. Cavallaro et al. extend their prior work with algorithms that ensure minimal feedback arc sets maintain graph centrality measures, using topological sorting to balance vertex degrees and enhance robustness in network analysis tasks [6]. Xiong et al. present reduction rules and divide-and-conquer strategies tailored for very large graphs, achieving near-optimal small feedback arc sets through kernelization techniques that significantly prune instance sizes before applying local search [21].

2 QuickFAST Algorithm

QuickFAST is our quick and simple randomized approximation algorithm for *FAST* problem whose pseudocode is depicted in Algorithm 1. As you can see, it is similar to KwikSort. This algorithm returns an approximate solution for the *FAST* problem in the expected running time of $O(|V|\log|V|)$. QuickFAST needs a rounding function that is a function $f : V \times V \to [0,1]$ such that $f(i,j) = 1 - f(j,i)$ for any pair of distinct $i, j \in V$. For brevity, we use f_{ij} instead of $f(i,j)$. If the tournament satisfies the probability constraints, we use the identity rounding function $f_{ij} = w_{ij}$, and if the tournament satisfies the triangle inequality constraints in addition to the probability constraints, we use rounding function $f_{ij} = h(w_{ij})$, where,

$$h(w) = \begin{cases} 0 & \text{if } w \in I_1 = [0, 1/3] \\ 3w - 1 & \text{if } w \in I_2 = [1/3, 2/3] \\ 1 & \text{if } w \in I_3 = [2/3, 1] \end{cases}$$

QuickFAST$((V, w))$**:**
  if $V = \emptyset$ **then**
    return an empty ranking;
  else
    Choose $p \in V$ uniformly at random;
    Intialize empty lists V_L and V_R;
    for *each* $u \in V - \{p\}$ **do**
      Add u to V_L with probability f_{up}, otherwise (with probability $f_{pu} = 1 - f_{up}$) add u to V_R;
    return Concatenation of **QuickFAST** $\big((V_L, w)\big)$, p, **QuickFAST** $\big((V_R, w)\big)$

Algorithm 1: Pseudocode of *QuickFAST*.

2.1 Analysis of the Algorithm

For any pairs of distinct vertices i and j, exactly one of the edges (i, j) or (j, i) is a backward edge in the ranking returned by *QuickFAST*. We call the backward edge between vertices i and j *good backward* if these vertices were in a recursive call of the algorithm, and one of them has been selected as a pivot in that call. A backward edge is *bad backward* if it is not good; in other words, the backward edge between vertices i and j is a *bad backward* if there exists a vertex k along with vertices i and j in a recursive call of the algorithm, and in this recursive call, k is selected as the pivot, and vertices i and j are placed in different sides of k.

We define $E^k_{ij} (= E^k_{ji})$ as the event in which vertices i, j, and k co-exist in a recursive call, and in that recursive call, k is chosen as the pivot. Events E^k_{ij}, E^i_{jk}, and E^j_{ki} have equal probabilities because we choose the pivot vertex uniformly at random. Let F^k_{ij} be the event that pivot k sets i to its left side and j to its right side. The events F^k_{ij} and $F^{k'}_{ij}$ are disjoint for any distinct pair $k, k' \in V - \{i, j\}$, because when a pivot vertex puts i and j to its different sides, i and j can no longer co-exist in another recursive call. Therefore, with probability $\Pr\left[\bigcup_{k \in V-\{i,j\}} F^k_{ij}\right] = \sum_{k \in V-\{i,j\}} \Pr\left[F^k_{ij}\right]$, the edge (j, i) is a bad backward edge. The event F^k_{ij} occurs when E^k_{ij} occurs and the pivot k sets i and j to its left and right sides, respectively. Therefore, $\Pr\left[F^k_{ij}\right] = \Pr\left[E^k_{ij}\right] D^k_{ij}$, where $D^k_{ij} := f_{ik} f_{kj}$. In fact, D^k_{ij} is the probability that while i, j and k are in a same recursion step and k is picked as the pivot of that step, i and j is set to the left and right of k, respectively.

Let π_{QF} be the ranking returned by *QuickFAST*, and random variables B_{QF} and G_{QF} be the sum of the weights of all bad and good backward edges in π_{QF}, respectively. This means that

$$E[cost(\pi_{QF})] = E[B_{QF}] + E[G_{QF}].$$

The expected value of B_{QF} is computed as follows.

$$E\big[B_{QF}\big] = \sum_{i<j} \sum_{k \in V-\{i,j\}} \Pr\left[E^k_{ij}\right] D^k_{ij} w_{ji} + \Pr\left[E^k_{ij}\right] D^k_{ji} w_{ij} = \sum_{i<j<k} \Pr\left[E^k_{ij}\right] \phi_{ijk} \tag{2}$$

where,

$$\phi_{ijk} = D^k_{ij} w_{ji} + D^k_{ji} w_{ij} + D^i_{jk} w_{kj} + D^i_{kj} w_{jk} + D^j_{ki} w_{ik} + D^j_{ik} w_{ki}.$$

The backward edge between i and j is bad with probability

$$\sum_{k\in V-\{i,j\}} \Pr[F_{ij}^k \cup F_{ji}^k] = \sum_{k\in V-\{i,j\}} \Pr[F_{ij}^k] + \Pr[F_{ji}^k] = \sum_{k\in V-\{i,j\}} \Pr[E_{ij}^k]\big(D_{ij}^k + D_{ji}^k\big).$$

So, the backward edge between i and j with probability $1 - \sum_{k\in V-\{i,j\}} E_{ij}^k\big(D_{ij}^k + D_{ji}^k\big)$ is good. Therefore, the expected value of G_{QF} is computed as follows.

$$E\big[G_{QF}\big] = \sum_{i<j} \Big(1 - \sum_{k\in V-\{i,j\}} \Pr\big[E_{ij}^k\big]\big(D_{ij}^k + D_{ji}^k\big)\Big)\big(f_{ij}w_{ji} + f_{ji}w_{ij}\big) \tag{3}$$

Let C_{LP} be the optimal value of LP 1 that is $\sum_{i<j} x_{ij}w_{ji} + x_{ji}w_{ij}$. Then,

$$\begin{aligned} C_{LP} =& \sum_{i<j} x_{ij}w_{ji} + x_{ji}w_{ij} \\ =& \sum_{i<j} \sum_{k\in V-\{i,j\}} \Pr\big[E_{ij}^k\big]\big(D_{ij}^k + D_{ji}^k\big)\big(x_{ij}w_{ji} + x_{ji}w_{ij}\big) + \\ & \sum_{i<j} \Big(1 - \sum_{k\in V-\{i,j\}} \Pr\big[E_{ij}^k\big]\big(D_{ij}^k + D_{ji}^k\big)\Big)\big(x_{ij}w_{ji} + x_{ji}w_{ij}\big). \end{aligned}$$

We define,

$$B_{LP} = \sum_{i<j} \sum_{k\in V-\{i,j\}} \Pr\big[E_{ij}^k\big]\big(D_{ij}^k + D_{ji}^k\big)\big(x_{ij}w_{ji} + x_{ji}w_{ij}\big) = \sum_{i<j<k} \Pr\big[E_{ij}^k\big]\psi_{ijk} \tag{4}$$

where,

$$\begin{aligned} \psi_{ijk} =& \big(D_{ij}^k + D_{ji}^k\big)(x_{ji}w_{ij} + x_{ij}w_{ji}) + \\ & \big(D_{jk}^i + D_{kj}^i\big)(x_{kj}w_{jk} + x_{jk}w_{kj}) + \\ & \big(D_{ki}^j + D_{ik}^j\big)(x_{ik}w_{ki} + x_{ki}w_{ik}) \end{aligned}$$

and

$$G_{LP} = \sum_{i<j} \Big(1 - \sum_{k\in V-\{i,j\}} \Pr\big[E_{ij}^k\big]\big(D_{ij}^k + D_{ji}^k\big)\Big)\big(x_{ij}w_{ji} + x_{ji}w_{ij}\big). \tag{5}$$

Then,

$$C_{LP} = B_{LP} + G_{LP}.$$

Let OPT be the *cost* of the optimal value of the *feedback arc set* problem. Then, C_{LP} is a lower bound for OPT [2].

Note that for any pair of distinct vertices a and b, the second constraint of LP 1 implies that $x_{ba} = 1 - x_{ab}$, and the probability constraints imply that $w_{ba} = 1 - w_{ab}$. If we use one of the rounding functions that we presented in Section 2 as the rounding function of *QuickFAST* (i.e. $f_{ij} = h(w_{ij})$ or $f_{ij} = w_{ij}$), then ϕ_{ijk} and ψ_{ijk} are functions with six variables x_{ij}, x_{jk}, x_{ki}, w_{ij}, w_{jk}, and w_{ki}.

Lemma 1. QuickFAST *is an α-approximation algorithm for the feedback arc set problem if the following two conditions hold:*

1. *For any pair of distinct vertices i and j,*

$$\Delta_{FAST}(x_{ij}, w_{ij}) := \big(f_{ij}w_{ji} + f_{ji}w_{ij}\big) - \alpha\big(x_{ij}w_{ji} + x_{ji}w_{ij}\big) \leq 0.$$

2. *For any triple of distinct vertices i, j, and k,*

$$\Omega_{FAST}(x_{ij}, x_{jk}, x_{ki}, w_{ij}, w_{jk}, w_{ki}) := \phi_{ijk} - \alpha \cdot \psi_{ijk} \leq 0.$$

Proof: If Condition 1 holds, then $E\big[G_{QF}\big] \leq \alpha \cdot G_{LP}$ is a direct consequence of equations 3 and 5. If Condition 2 holds, then from equations 2 and 4, we have $E\big[B_{QF}\big] \leq \alpha \cdot B_{LP}$. Therefore,

$$E\big[cost(\pi_{QF})\big] = E\big[B_{QF}\big] + E\big[G_{QF}\big] \leq \alpha \cdot B_{LP} + \alpha \cdot G_{LP} = \alpha \cdot C_{LP} \leq \alpha \cdot OPT.$$

□

To prove an approximation factor α for QuickFAST using Lemma 1, we need to show that the maximum values of Δ_{FAST} and Ω_{FAST} are less than or equal to 0 for that value of α.

Theorem 1. *The QuickFAST algorithm with $f_{ij} = w_{ij}$ is a 3-approximation algorithm for the feedback arc set problem on tournaments that satisfy probability constraints.*

Proof: We will show that both conditions of Lemma 1 hold for $\alpha = 3$. Without loss of generality, suppose that $w_{ij} \geq w_{ji}$.

$$\begin{aligned}
\Delta_{FAST}(x_{ij}, w_{ij}) &= f_{ij}w_{ji} + f_{ji}w_{ij} - \alpha(x_{ij}w_{ji} + x_{ji}w_{ij}) \\
&\leq w_{ij}w_{ji} + w_{ji}w_{ij} - 3(x_{ij}w_{ji} + x_{ji}w_{ji}) && \text{(Since } w_{ij} \geq w_{ji}.\text{)} \\
&= 2w_{ji}w_{ij} - 3w_{ji} && \text{(Since } x_{ij} + x_{ji} = 1.\text{)} \\
&\leq 2w_{ji} - 3w_{ji} && \text{(Since } 0 \leq w_{ij} \leq 1.\text{)} \\
&\leq 0.
\end{aligned}$$

Therefore, Condition 1 holds for $\alpha = 3$. By the definitions of ϕ_{ijk} and ψ_{ijk}, we have,

$$\phi_{ijk} = 3(w_{ik}w_{kj}w_{ji} + w_{ij}w_{jk}w_{ki}) \tag{6}$$

and

$$\begin{aligned}
\psi_{ijk} \geq &(x_{ji}w_{ij}D^k_{ji} + x_{ij}w_{ji}D^k_{ij}) + \\
&(x_{kj}w_{jk}D^i_{kj} + x_{jk}w_{kj}D^i_{jk}) + \\
&(x_{ik}w_{ki}D^j_{ik} + x_{ki}w_{ik}D^j_{ki}) \\
=&(x_{ij} + x_{jk} + x_{ki})(w_{ik}w_{kj}w_{ji}) + (x_{ik} + x_{kj} + x_{ji})(w_{ij}w_{jk}w_{ki}) \\
\geq &w_{ik}w_{kj}w_{ji} + w_{ij}w_{jk}w_{ki}.
\end{aligned} \tag{7}$$

The last equality follows from the first condition of LP 1. By Equality 6 and Inequality 7, we have

$$\Omega_{FAST}(x_{ij}, x_{jk}, x_{ki}, w_{ij}, w_{jk}, w_{ki}) = \phi_{ijk} - 3\psi_{ijk} \leq 0.$$

Therefore, Condition 2 holds for $\alpha = 3$, as well. □

Theorem 2 gives the approximation factor for our algorithm on tournaments satisfying the probability and triangle inequality constraints. Our proof hinges on two following lemmas, stated below. We first present the proof of Theorem 2 using these lemmas, and subsequently prove the lemmas themselves.

Lemma 2. *If $f_{ij} = h(w_{ij})$ in QuickFAST algorithm, the first condition of Lemma 1 holds for $\alpha = 5/3$ on tournaments that satisfy both the probability and the triangle inequality constraints.*

Lemma 3. *If $f_{ij} = h(w_{ij})$ in QuickFAST algorithm, then the second condition of Lemma 1 holds for $\alpha = 5/3$ if the input tournament satisfies both the probability and the triangle inequality constraints.*

Theorem 2. *The QuickFAST algorithm with $f_{ij} = h(w_{ij})$ is a 5/3-approximation algorithm for the feedback arc set problem on tournaments that satisfy both probability and triangle inequality constraints.*

Proof: By Lemma 2 and Lemma 3, two conditions of Lemma 1 hold for $\alpha = 5/3$ which complete the proof. □

In the rest of this section, we will prove Lemma 2 and Lemma 3. To do this, for each of the functions Ω_{FAST} and Δ_{FAST}, we will find a subset of its domain such that the maximum value of the function on its domain and on that subset are the same. This helps us to find the maximum values more simply.

Proof: [Lemma 2] In Δ_{FAST}, we set f_{ij} to $h(w_{ij})$, if tournament satisfies both probability and triangle inequality constraints. If we fix the value of w_{ij} then Δ_{FAST} is a linear function with single variable x_{ij}, and it can reach to its maximum value when x_{ij} is in $\{0, 1\}$, which is the set of the endpoints of the domain of x_{ij}. On the other hand, $\Delta_{FAST}(1, a) = \Delta_{FAST}(0, 1 - a)$ for any $a \in [0, 1]$. So, instead of the whole domain of Δ_{FAST}, we can find the maximum value of Δ on $D(\Delta) := \big\{(0, w_{ij}) \,\big|\, w_{ij} \in [0, 1]\big\}$. We will show that $\Delta_{FAST} \leq 0$ on domain $D(\Delta)$ for $\alpha = 5/3$.

- If $0 \leq w_{ij} \leq 1/3$ then,

$$\begin{aligned}\Delta_{FAST}(0, w_{ij}) &= \big(f_{ij}w_{ji} + f_{ji}w_{ij}\big) - \alpha\big(x_{ij}w_{ji} + x_{ji}w_{ij}\big)\\ &= \big(0 \times (1 - w_{ij}) + 1 \times w_{ij}\big) - \frac{5}{3}\big(0 \times (1 - w_{ij}) + 1 \times w_{ij}\big)\\ &= w_{ij} - \frac{5}{3}w_{ij} \leq 0.\end{aligned}$$

- If $1/3 \leq w_{ij} \leq 2/3$ then,

$$\begin{aligned}\Delta_{FAST}(0, w_{ij}) &= \big(3w_{ij} - 1\big)\big(1 - w_{ij}\big) + \big(3(1 - w_{ij}) - 1\big)w_{ij} - \frac{5}{3}w_{ij}\\ &= -6w_{ij}^2 + \frac{13}{3}w_{ij} - 1 < 0.\end{aligned}$$

- If $2/3 \leq w_{ij} \leq 1$ then,

$$\Delta_{FAST}(0, w_{ij}) = 1 \times \big(1 - w_{ij}\big) + 0 \times w_{ij} - \frac{5}{3}w_{ij} = 1 - \frac{8}{3}w_{ij} \leq 0.$$

□

Proof: [Lemma 3] By the first constraint of LP 1, For any $i, j, k \in V$, we have $1 \leq x_{ij} + x_{jk} + x_{ki}$, and also $1 \leq x_{kj} + x_{ji} + x_{ik}$. Therefore,

$$1 \leq x_{kj} + x_{ji} + x_{ik} = (1 - x_{jk}) + (1 - x_{ij}) + (1 - x_{ki}) \implies x_{ij} + x_{jk} + x_{ki} \leq 2.$$

Thus, the domain of (x_{ij}, x_{jk}, x_{ki}) is

$$U = \{(a,b,c) \,\big|\, a,b,c \in [0,1] \wedge 1 \leq a+b+c \leq 2\}.$$

Similarly, we can show that the domain of (w_{ij}, w_{jk}, w_{ki}) is U, if the tournament satisfies both the probability and the triangle inequality constraints.

If in Ω_{FAST}, we set f_{ij} to one of the rounding functions which we defined, and fix the values of variables w_{ij}, w_{jk} and w_{ki} then Ω_{FAST} is a linear function with three variables x_{ij}, x_{jk} and x_{ki}. Therefore, Ω_{FAST} will receive its maximum value when (x_{ij}, x_{jk}, x_{ki}) is in the set of vertices of its domain that is the polytope U. These vertices are $(0,0,1)$, $(0,1,0)$, $(1,0,0)$, $(0,1,1)$, $(1,0,1)$ and $(1,1,0)$. On the other hand, for any $a,b,c \in [0,1]$, due to the symmetry of Ω_{FAST}, we have

$$\begin{aligned}&\Omega_{FAST}(1,0,0,a,b,c) = \Omega_{FAST}(0,1,0,c,a,b) = \Omega_{FAST}(0,0,1,b,c,a) = \\ &\Omega_{FAST}(0,1,1,1-a,1-b,1-c) = \Omega_{FAST}(1,0,1,1-c,1-a,1-b) = \\ &\Omega_{FAST}(1,1,0,1-b,1-c,1-a).\end{aligned}$$

So, to find the maximum value of Ω_{FAST}, we can only focus on $(x_{ij}, x_{jk}, x_{ki}) = (1,1,0)$.

For any $a,b,c \in \{1,2,3\}$ we define

$$P_{abc} = \{(1,1,0,w_{ij},w_{jk},w_{ki}) \mid (w_{ij},w_{jk},w_{ki}) \in I_a \times I_b \times I_c \cap U\}$$

where I_1, I_2, and I_3 are the intervals which are defined in the definition of function h. Thus, for tournaments that satisfy both the probability and the triangle inequality constraints, it is enough to find the maximum value of Ω_{FAST} on $\bigcup_{a,b,c} P_{abc}$. In these tournaments, $f_{ij} = h(w_{ij})$. So, if we fix all variables of Ω_{FAST} except one from w_{ij}, w_{jk} or w_{ki}, then Ω_{FAST} on a specific P_{abc} is a line (in degenerate case, a point). Hence, for such a situation with only one non-fix argument $w' \in \{w'_{ij}, w'_{jk}, w'_{ki}\}$, the value of Ω_{FAST} decreases in at most one direction of changes of w' (when we decrease or increase the value of w' it is impossible that the value of Ω_{FAST} getting decreased in both directions of changes of w'). Thus, Ω_{FAST} on a specific P_{abc} receives its maximum value in one of the vertices of $I_a \times I_b \times I_c$ or in a point at the boundary of U. So, there is a point that maximizes Ω_{FAST} and has at least one of the following conditions:

- $w_{ij} + w_{jk} + w_{ki} \in \{1,2\}$
- $w_{ij}, w_{jk}, w_{ki} \in \{0, 1/3, 2/3, 1\}$

Therefore, when tournaments satisfy both the probability and the triangle inequality constraints, its enough to find the maximum value of Ω_{FAST} on

$$D_T(\Omega) = \big\{(1,1,0,w_{ij},w_{jk},w_{ki}) \,\big|\, (w_{ij},w_{jk},w_{ki}) \in Y\big\}$$

where,

$$Y = \big\{(a,b,c) \,\big|\, (a,b,c) \in U \wedge \big(a+b+c \in \{1,2\} \vee a,b,c \in \{0,\frac{1}{3},\frac{2}{3},1\}\big)\big\}.$$

Let $w_{ij} + w_{jk} + w_{ki} \in \{1,2\}$. As $w_{ki} \in [0,1]$, if $w_{ij} + w_{jk} < 1$, then $w_{ki} = 1 - w_{ij} - w_{jk}$; if $w_{ij} + w_{jk} > 1$, then $w_{ki} = 2 - w_{ij} - w_{jk}$; and if $w_{ij} + w_{jk} = 1$, then $w_{ki} = 1 - w_{ij} - w_{jk}$

or $w_{ki} = 2 - w_{ij} - w_{jk}$. Besides that, $\Omega(1,1,0,w_{ij},w_{jk},w_{ki}) = \Omega(1,1,0,w_{jk},w_{ij},w_{ki})$ due to symmetry. Therefore, $D_T(\Omega) = D_1 \cup D_2 \cup D_3$ where,

$$D_1 = \big\{(1,1,0,w_{ij},w_{jk},1-w_{ij}-w_{jk}) \;\big|\; w_{ij},w_{jk} \in [0,1],\, w_{ij} \le w_{jk},\, w_{ij}+w_{jk} \le 1\big\}$$
$$D_2 = \big\{(1,1,0,w_{ij},w_{jk},2-w_{ij}-w_{jk}) \;\big|\; w_{ij},w_{jk} \in [0,1],\, w_{ij} \le w_{jk},\, w_{ij}+w_{jk} \ge 1\big\}$$
$$D_3 = \big\{(1,1,0,w_{ij},w_{jk},w_{ki}) \;\big|\; (w_{ij},w_{jk},w_{ki}) \in U,\, w_{ij} \le w_{jk},\, w_{ij},w_{jk},w_{ki} \in \{0,1/3,2/3,1\}\big\}$$

All possible values of D_3 and their corresponding values for Ω_{FAST} have been shown in Table 2, which shows that $\Omega_{FAST} \le 0$ on D_3. We will complete the formal proof for this lemma by giving a detailed discussion in Appendix A showing that the maximum value of Ω_{FAST} on $D_1 \cup D_2$ is also less than or equal to 0. □

3 QuickRank Algorithm

For two rankings π_1 and π_2 on a set V, the *Kendall-Tau* distance is defined as,

$$K(\pi_1,\pi_2) = \big|\big\{(i,j) \in (V \times V)\ s.t.\ \pi_1(i) < \pi_1(j) \wedge \pi_2(i) > \pi_2(j)\big\}\big|.$$

Having a set of rankings $\pi_1,\dots,\pi_m$ on V, Kemeny gave a measure for the best ranking that aggregates these rankings [12]: the *rank aggregation* problem (Kemeny optimal ranking) is to find a ranking π_{Rank} whose sum of the *Kendall-Tau* distances from $\pi_1,\dots,\pi_m$ is minimum. In other words,

$$\pi_{Rank} = \operatorname*{argmin}_{\pi} \sum_{t=1}^{m} K(\pi,\pi_t).$$

The *rank aggregation* problem is NP-hard even if there are only four rankings [4, 9].

This problem can be reduced to the *minimum feedback arc set* problem as follows. Assume that rankings $\pi_1,\dots,\pi_m$ on set V of size n is an instance of the *rank aggregation* problem. We build a tournament (V,w) based on these rankings, denoted by $T(\pi_1,\dots,\pi_m)$, where $w_{ij} = \frac{1}{m}|\{t \in \{1,\dots,m\}\ s.t.\ \pi_t(i) < \pi_t(j)\}|$. Then a π_{FAST} for (V,w) is a π_{Rank} for $T(\pi_1,\dots,\pi_m)$.

The tournament $T(\pi_1,\dots,\pi_m)$ satisfies the probability constraints:

$$w_{ij} + w_{ji} = \frac{|\{t \in \{1,\dots,m\}\ s.t.\ \pi_t(i) < \pi_t(j)\}|}{m} + \frac{|\{t \in \{1,\dots,m\}\ s.t.\ \pi_t(j) < \pi_t(i)\}|}{m} = 1.$$

Moreover, $T(\pi_1,\dots,\pi_m)$ satisfies the triangle inequality constraints: for any triple of distinct $i,j,k \in V$, and $t \in \{1,\dots,m\}$ if $\pi_t(i) < \pi_t(k)$ then $\pi_t(i) < \pi_t(j)$ or $\pi_t(j) < \pi_t(k)$. Therefore,

$$\begin{aligned} w_{ik} &= \frac{|\{t \in \{1,\dots,m\}\ s.t.\ \pi_t(i) < \pi_t(k)\}|}{m} \\ &\le \frac{|\{t \in \{1,\dots,m\}\ s.t.\ \pi_t(i) < \pi_t(j)\}|}{m} + \frac{|\{t \in \{1,\dots,m\}\ s.t.\ \pi_t(j) < \pi_t(k)\}|}{m} \\ &= w_{ij} + w_{jk}. \end{aligned}$$

Pick-A-Perm is a trivial approximation algorithm for *rank aggregation* problem whose running time is $O(1)$ [2]. This algorithm simply returns one of the input rankings uniformly at random. Let π_{PAP} be the output of Pick-A-Perm and random variable Z_{ij} be the weight of the backward edge between vertices i and j in π_{PAP}. If $\pi_{PAP}(i) < \pi_{PAP}(j)$, the edge (j,i) is backward and

$Z_{ij} = w_{ji}$. This event happens with probability $\frac{1}{m}|\{t \in \{1,\dots,m\}\ s.t.\ \pi_t(i) < \pi_t(j)\}| = w_{ij}$. Similarly, $Z_{ij} = w_{ij}$ with probability w_{ji}. So, $E[Z_{ij}] = 2w_{ij}w_{ji}$ and

$$E[cost(\pi_{PAP})] = \sum_{i<j} E[Z_{ij}] = \sum_{i<j} 2w_{ij}w_{ji}. \tag{8}$$

In π_{Rank}, for any pair of distinct vertices i and j, one of the edges (i,j) or (j,i) is backward. Hence, $\sum_{i<j} \min(w_{ij}, w_{ji})$ is a lower bound for $cost(\pi_{Rank})$. On the other hand, $w_{ij}w_{ji} \le \min(w_{ij}, w_{ji})$ because $w_{ij}, w_{ji} \in [0,1]$. Therefore,

$$E[cost(\pi_{PAP})] = \sum_{i<j} 2w_{ij}w_{ji} = 2\sum_{i<j} \min(w_{ij}, w_{ji}) \le 2 \cdot cost(\pi_{Rank}).$$

This implies that the approximation factor of Pick-A-Perm is 2 for the *rank aggregation* problem. Trivially, KwikSort$(T(\pi_1,\dots,\pi_m))$ is another 2-approximation algorithm for this problem. Selecting the best solution among Pick-A-Perm$(\pi_1,\dots,\pi_m)$ and KwikSort$(T(\pi_1,\dots,\pi_m))$ is a 11/7-approximation algorithm [2] for the *rank aggregation* of $\pi_1,\dots,\pi_m$.

As QuickFAST improves the efficiency of KwikSort for *FAST* problem, it is natural to expect the same improvement by using the combination of Pick-A-Perm and QuickFAST for *rank aggregation* problem. Our proposed algorithm for this combination, called QuickRank, is simple and fast: With probability β it returns the result of QuickFAST, and otherwise (with probability $1-\beta$) returns Pick-A-Perm. We need $O(m)$ time to find the weight of any edge of $T(\pi_1, \pi_2, \cdots, \pi_m)$, but we do not need to compute the weights of all edges because in each level of recursion tree at QuickFAST we only need the weights of $O(V)$ edges. The expected value of recursion levels is $O(\log|V|)$. So, the expected running time of QuickRank is $O(m|V|\log|V|)$. Instead of using randomness, we can select the best solution from Pick-A-Perm and QuickFAST. This approach achieves an approximation factor at least as good as that of QuickRank.

Now we analyze the approximation factor of this algorithm. From equation 8, we have

$$\begin{aligned} E[cost(\pi_{PAP})] &= \sum_{i<j} 2w_{ij}w_{ji} \\ &= \sum_{i<j} \sum_{k\in V-\{i,j\}} \Pr\left[E_{ij}^k\right]\left(D_{ij}^k + D_{ji}^k\right)\left(2w_{ij}w_{ji}\right) + \\ &\quad \sum_{i<j} \Big(1 - \sum_{k\in V-\{i,j\}} \Pr\left[E_{ij}^k\right]\left(D_{ij}^k + D_{ji}^k\right)\Big)\left(2w_{ij}w_{ji}\right). \end{aligned}$$

We define,

$$B_{PAP} = \sum_{i<j} \sum_{k\in V-\{i,j\}} \Pr\left[E_{ij}^k\right]\left(D_{ij}^k + D_{ji}^k\right)\left(2w_{ij}w_{ji}\right) = \sum_{i<j<k} \Pr\left[E_{ij}^k\right]\mu_{ijk} \tag{9}$$

where,

$$\mu_{ijk} = \left(D_{ij}^k + D_{ji}^k\right)(2w_{ij}w_{ji}) + \left(D_{jk}^i + D_{kj}^i\right)(2w_{jk}w_{kj}) + \left(D_{ki}^j + D_{ik}^j\right)(2w_{ki}w_{ik})$$

and

$$G_{PAP} = \sum_{i<j} \Big(1 - \sum_{k\in V-\{i,j\}} \Pr\left[E_{ij}^k\right]\left(D_{ij}^k + D_{ji}^k\right)\Big)\left(2w_{ij}w_{ji}\right). \tag{10}$$

Then,

$$E[cost(\pi_{PAP})] = B_{PAP} + G_{PAP}.$$

Lemma 4. QuickRank *is an α-approximation algorithm for the rank aggregation problem if the following two conditions hold for some $\beta \in [0,1]$:*

1. *For any pair of distinct vertices i and j,*

$$\Delta_{Rank}(x_{ij}, w_{ij}) = \big(\beta(f_{ij}w_{ji} + f_{ji}w_{ij}) + (1-\beta)(2w_{ij}w_{ji})\big) - \alpha\big(x_{ij}w_{ji} + x_{ji}w_{ij}\big) \leq 0.$$

2. *For any triple of distinct vertices i, j, and k,*

$$\Omega_{Rank}(x_{ij}, x_{jk}, x_{ki}, w_{ij}, w_{jk}, w_{ki}) = \big(\beta \cdot \phi_{ijk} + (1-\beta)\mu_{ijk}\big) - \alpha \cdot \psi_{ijk} \leq 0.$$

Proof: Let π_{QR} be the ranking which is returned by QuickRank. This algorithm with probability β is QuickFAST and with probability $1-\beta$ is Pick-A-Perm. So,

$$E\big[cost(\pi_{QR})\big] = \beta \cdot E\big[cost(\pi_{QF})\big] + (1-\beta)E\big[cost(\pi_{PAP})\big].$$

If Condition 1 holds, then from equations 3, 10, and 5, we have

$$\beta \cdot E\big[G_{QF}\big] + (1-\beta)E\big[G_{PAP}\big] \leq \alpha \cdot G_{LP}.$$

If Condition 2 holds, then from equations 2, 9, and 4, we have

$$\beta \cdot E\big[B_{QF}\big] + (1-\beta)E\big[B_{PAP}\big] \leq \alpha \cdot B_{LP}.$$

Therefore,

$$\begin{aligned} E\big[cost(\pi_{QR})\big] &= \beta \cdot E\big[cost(\pi_{QF})\big] + (1-\beta)E\big[cost(\pi_{PAP})\big] \\ &= \beta\big(E\big[B_{QF}\big] + E\big[G_{QF}\big]\big) + (1-\beta)\big(E\big[B_{PAP}\big] + E\big[G_{PAP}\big]\big) \\ &= \beta \cdot E\big[B_{QF}\big] + (1-\beta)E\big[B_{PAP}\big] + \beta \cdot E\big[G_{QF}\big] + (1-\beta)E\big[G_{PAP}\big] \\ &\leq \alpha \cdot B_{LP} + \alpha \cdot G_{LP} = \alpha \cdot C_{LP} \leq \alpha \cdot OPT. \end{aligned}$$

□

By the same proof as the proof of Lemma 2, it can be concluded that if in *QuickRank* algorithm $f_{ij} = h(w_{ij})$, the first condition of Lemma 4 holds for $\alpha = 1.45$, and $\beta = 0.55$. But to prove the second condition of Lemma 4 like the formal proof of Lemma 3, we must provide a restriction for the domain of Ω_{Rank} like what we did for the domain of Ω_{FAST}. However, because of the more algebraic complexity of this function it is too complicated to do this. We informally, checked the correctness of this condition using a computer program and checking all points on a sufficiently fine-grained grid. So, we claim the result as a conjecture.

Conjecture 1. *QuickRank algorithm with $\beta = 0.55$ is a 1.45-approximation algorithm for the rank aggregation problem.*

While the focus of this work is on establishing theoretical bounds for worst-case inputs, we also provide an open-source benchmarking suite. The implementation, which generates random tournaments and compares the average-case empirical performance of QuickFAST against existing algorithms, is publicly accessible at: https://github.com/mojtabaOstovari/FAST-Benchmarking.

References

[1] N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. *Algorithmica*, 57(2):284–300, 2010. doi:10.1007/S00453-008-9211-1.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5), 2008. doi:10.1145/1411509.1411513.

[3] N. Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006. arXiv:https://doi.org/10.1137/050623905, doi:10.1137/050623905.

[4] J. Bartholdi, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989. URL: http://www.jstor.org/stable/41105913.

[5] M. Braverman and E. Mossel. Noisy Sorting Without Resampling. *arXiv e-prints*, page arXiv:0707.1051, July 2007. arXiv:0707.1051, doi:10.48550/arXiv.0707.1051.

[6] C. Cavallaro, V. Cutello, and M. Pavone. Efficient heuristics to compute minimal and stable feedback arc sets. *J. Comb. Optim.*, 48(4), Oct. 2024. doi:10.1007/s10878-024-01209-8.

[7] D. Coppersmith, L. K. Fleischer, and A. Rurda. Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Transactions on Algorithms*, 6(3), 2010. doi:10.1145/1798596.1798608.

[8] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, editors, *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 613–622. ACM, 2001. doi:10.1145/371920.372165.

[9] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 613–622, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/371920.372165.

[10] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM International Conference on Management of Data (SIGMOD)*, page 301–312. Association for Computing Machinery, 2003. doi:10.1145/872757.872795.

[11] H. Jung. On subgraphs without cycles in tournaments. *Combinatorial Theory and its Applications II*, pages 675–677, 1970.

[12] J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959. URL: http://www.jstor.org/stable/20026529.

[13] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 95–103. ACM, 2007. doi:10.1145/1250790.1250806.

[14] M. Ostovari and A. Zarei. A better LP rounding for feedback arc set on tournaments. *Theoretical Computer Science*, 1015:114768, 2024. doi:10.1016/j.tcs.2024.114768.

[15] K. Reid. On sets of arcs containing no cycles in a tournament. *Canadian Mathematical Bulletin*, 12(3):261–264, 1969. doi:10.4153/CMB-1969-032-x.

[16] R. Remage and W. A. Thompson. Maximum-likelihood paired comparison rankings. *Biometrika*, 53(1/2):143–149, 1966. URL: http://www.jstor.org/stable/2334060.

[17] M. Simpson, V. Srinivasan, and A. Thomo. Efficient computation of feedback arc set at web-scale. *Proc. VLDB Endow.*, 10(3):133–144, Nov. 2016. doi:10.14778/3021924.3021930.

[18] P. Slater. Inconsistencies in a schedule of paired comparisons. *Biometrika*, 48:303–312, 1961. URL: https://api.semanticscholar.org/CorpusID:123662615.

[19] J. Spencer. Optimally ranking unrankable tournaments. *Periodica Mathematica Hungarica*, 11(2):131 – 144, 1980. doi:10.1007/bf02017965.

[20] A. van Zuylen and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. volume 34, pages 594–620, 2009. URL: http://www.jstor.org/stable/40538434.

[21] Z. Xiong, Y. Zhou, M. Xiao, and B. Khoussainov. Finding small feedback arc sets on large graphs. *Computers & Operations Research*, 169:106724, 2024. URL: https://www.sciencedirect.com/science/article/pii/S0305054824001965, doi:10.1016/j.cor.2024.106724.

A Proof of Lemma 3

We will show that the maximum value of Ω_{FAST} on $D_T(\Omega) = D_1 \cup D_2 \cup D_3$ is less than or equal to 0. Table 2 shows the values of Ω_{FAST} on D_3 which are less than or equal to 0. We used function h as the rounding function. Thus Ω_{FAST} is a piecewise function, and we need to compute the maximum value on each piece separately on D_1 and D_2. In the following, you can see the maximum value of Ω_{FAST} on D_1 and D_2 when $(w_{ij}, w_{jk}, w_{ki}) \in I_a \times I_b \times I_c$ for any $a, b, c \in \{1, 2, 3\}$.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_1 \times I_1$:
  1. On D_1:
  $$\Omega_{FAST} = \frac{10w_{ij}}{3} + \frac{10w_{jk}}{3} - 3 \qquad \text{argmax} = (\frac{1}{3}, \frac{1}{3}) \quad \max = -\frac{7}{9}$$
  2. On D_2: There is no point with these conditions.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_1 \times I_2$:
  1. On D_1:
  $$\Omega_{FAST} = 2w_{ij}^2 + 4w_{ij}w_{jk} - 2w_{ij} + 2w_{jk}^2 - 2w_{jk} - \frac{1}{3} \quad \text{argmax} = (0, \frac{1}{3}) \quad \max = -\frac{7}{9}$$
  2. On D_2: There is no point with these conditions.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_1 \times I_3$:

1. On D_1:

$$\Omega_{FAST} = \frac{8w_{ij}}{3} + \frac{8w_{jk}}{3} - \frac{5}{3} \qquad \text{argmax} = (0, \frac{1}{3}) \quad \text{max} = -\frac{7}{9}$$

2. On D_2: There is no point with these conditions.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_2 \times I_1$:

1. On D_1:

$$\Omega_{FAST} = -10w_{ij}w_{jk} + \frac{20w_{ij}}{3} - 8w_{jk}^2 + 13w_{jk} - \frac{16}{3}$$
$$\text{argmax} = (\frac{1}{3}, \frac{29}{48}) \quad \text{max} = -\frac{55}{288}$$

2. On D_2: There is no point with these conditions.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_2 \times I_2$:

1. On D_1:

$$\Omega_{FAST} = -30w_{ij}^2w_{jk} + 12w_{ij}^2 - 30w_{ij}w_{jk}^2 + 45w_{ij}w_{jk} - \frac{37w_{ij}}{3} + 15w_{jk}^2 - \frac{40w_{jk}}{3} + 2$$
$$\text{argmax} = (0, \frac{2}{3}) \quad \text{max} = -\frac{2}{9}$$

2. On D_2: There is no point with these conditions.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_2 \times I_3$:

1. On D_1:

$$\Omega_{FAST} = \frac{8w_{ij}}{3} - 8w_{jk}^2 + \frac{16w_{jk}}{3} - \frac{5}{3} \qquad \text{argmax} = (0, \frac{1}{3}) \quad \text{max} = -\frac{7}{9}$$

2. On D_2:

$$\Omega_{FAST} = \frac{8w_{ij}}{3} - 8w_{jk}^2 + \frac{40w_{jk}}{3} - 7 \qquad \text{argmax} = (\frac{1}{3}, \frac{2}{3}) \quad \text{max} = -\frac{7}{9}$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_3 \times I_1$:

1. On D_1:

$$\Omega_{FAST} = \frac{2w_{jk}}{3} - \frac{2}{3} \qquad \text{argmax} = (0, 1) \quad \text{max} = 0$$

2. On D_2: There is no point with these conditions.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_3 \times I_2$:

1. On D_1:

$$\Omega_{FAST} = -8w_{ij}^2 - 6w_{ij}w_{jk} + \frac{25w_{ij}}{3} + 2w_{jk}^2 + \frac{7w_{jk}}{3} - \frac{8}{3}$$
$$\text{argmax} = (0, \frac{2}{3}) \quad \max = -\frac{2}{9}$$

2. On D_2:

$$\Omega_{FAST} = -8w_{ij}^2 - 6w_{ij}w_{jk} + \frac{49w_{ij}}{3} + 2w_{jk}^2 + \frac{w_{jk}}{3} - \frac{17}{3}$$
$$\text{argmax} = (\frac{1}{3}, 1) \quad \max = -\frac{7}{9}$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_1 \times I_3 \times I_3$:

1. On D_1: There is no point with these conditions.
2. On D_2:

$$\Omega_{FAST} = \frac{8w_{ij}}{3} - \frac{5}{3} \qquad \text{argmax} = (\frac{1}{3}, 1) \quad \max = -\frac{7}{9}$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_2 \times I_2 \times I_1$:

1. On D_1:

$$\Omega_{FAST} = 30w_{ij}^2w_{jk} - 18w_{ij}^2 + 30w_{ij}w_{jk}^2 - 61w_{ij}w_{jk} + \frac{80w_{ij}}{3} - 18w_{jk}^2 + \frac{80w_{jk}}{3} - 10$$
$$\text{argmax} = (\frac{1}{2}, \frac{1}{2}) \quad \max = -\frac{1}{12}$$

2. On D_2: There is no point with these conditions.

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_2 \times I_2 \times I_2$:

1. On D_1:

$$\Omega_{FAST} = -30w_{ij}^2w_{jk} + 15w_{ij}^2 - 30w_{ij}w_{jk}^2 + 45w_{ij}w_{jk} - \frac{40w_{ij}}{3} + 15w_{jk}^2 - \frac{40w_{jk}}{3} + 2$$
$$\text{argmax} = (\frac{1}{3}, \frac{1}{3}) \quad \max = -\frac{7}{9}$$

2. On D_2:

$$\Omega_{FAST} = -30w_{ij}^2w_{jk} + 15w_{ij}^2 - 30w_{ij}w_{jk}^2 + 75w_{ij}w_{jk} - \frac{85w_{ij}}{3} + 15w_{jk}^2 - \frac{85w_{jk}}{3} + \frac{26}{3}$$
$$\text{argmax} = (\frac{2}{3}, \frac{2}{3}) \quad \max = -\frac{2}{9}$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_2 \times I_2 \times I_3$:

1. On D_1: There is no point with these conditions.
2. On D_2:

$$\Omega_{FAST} = 30w_{ij}^2w_{jk} - 18w_{ij}^2 + 30w_{ij}w_{jk}^2 - 71w_{ij}w_{jk} + \frac{101w_{ij}}{3} - 18w_{jk}^2 + \frac{101w_{jk}}{3} - \frac{46}{3}$$
$$\text{argmax} = (\frac{2}{3}, \frac{2}{3}) \quad \max = -\frac{2}{9}$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_2 \times I_3 \times I_1$:
  1. On D_1:
  $$\Omega_{FAST} = 2w_{ij}^2 - \frac{2w_{ij}}{3} + \frac{2w_{jk}}{3} - \frac{2}{3} \qquad \text{argmax} = (\frac{1}{3}, \frac{2}{3}) \quad \max = -\frac{2}{9}$$
  2. On D_2:
  $$\Omega_{FAST} = 2w_{ij}^2 - \frac{8w_{ij}}{3} + \frac{2w_{jk}}{3} \qquad \text{argmax} = (\frac{2}{3}, 1) \quad \max = -\frac{2}{9}$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_2 \times I_3 \times I_2$:
  1. On D_1: There is no point with these conditions.
  2. On D_2:
  $$\Omega_{FAST} = -30w_{ij}^2w_{jk} + 15w_{ij}^2 - 30w_{ij}w_{jk}^2 + 75w_{ij}w_{jk} - \frac{85w_{ij}}{3} + 12w_{jk}^2 - \frac{70w_{jk}}{3} + \frac{20}{3}$$
  $$\text{argmax} = (\frac{2}{3}, \frac{5}{6}) \quad \max = 0$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_2 \times I_3 \times I_3$:
  1. On D_1: There is no point with these conditions.
  2. On D_2:
  $$\Omega_{FAST} = 2w_{ij}^2 + 10w_{ij}w_{jk} - 7w_{ij} - \frac{10w_{jk}}{3} + \frac{4}{3} \quad \text{argmax} = (\frac{29}{48}, \frac{35}{48}) \quad \max = -\frac{55}{288}$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_3 \times I_3 \times I_1$:
  1. On D_1: There is no point with these conditions.
  2. On D_2:
  $$\Omega_{FAST} = \frac{2w_{ij}}{3} + \frac{2w_{jk}}{3} - \frac{4}{3} \qquad \text{argmax} = (1, 1) \quad \max = 0$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_3 \times I_3 \times I_2$:
  1. On D_1: There is no point with these conditions.

2. On D_2:

$$\Omega_{FAST} = -8w_{ij}^2 - 16w_{ij}w_{jk} + 24w_{ij} - 8w_{jk}^2 + 24w_{jk} - 18$$
$$\text{argmax} = (\frac{3}{2} - w_{jk}, w_{jk}) \quad \max = 0$$

- If $(w_{ij}, w_{jk}, w_{ki}) \in I_3 \times I_3 \times I_3$:
  1. On D_1: There is no point with these conditions.
  2. On D_2:

$$\Omega_{FAST} = \frac{10w_{ij}}{3} + \frac{10w_{jk}}{3} - \frac{14}{3} \qquad \text{argmax} = (\frac{2}{3}, \frac{2}{3}) \quad \max = -\frac{2}{9}$$

(w_{ij}, w_{jk}, w_{ki})	Ω_{FAST}
$(0, 0, 1)$	-5/3
$(0, 1/3, 2/3)$	-7/9
$(0, 1/3, 1)$	-5/3
$(0, 2/3, 1/3)$	-2/9
$(0, 2/3, 2/3)$	-5/3
$(0, 2/3, 1)$	-5/3
$(0, 1, 0)$	0
$(0, 1, 1/3)$	0
$(0, 1, 2/3)$	-5/3
$(0, 1, 1)$	-5/3
$(1/3, 1/3, 1/3)$	-7/9
$(1/3, 1/3, 2/3)$	-7/9
$(1/3, 1/3, 1)$	-5/3
$(1/3, 2/3, 0)$	-2/9
$(1/3, 2/3, 1/3)$	-2/9
$(1/3, 2/3, 2/3)$	-7/9
$(1/3, 2/3, 1)$	-7/9
$(1/3, 1, 0)$	0
$(1/3, 1, 1/3)$	0
$(1/3, 1, 2/3)$	-7/9
$(2/3, 2/3, 0)$	0
$(2/3, 2/3, 1/3)$	-2/9
$(2/3, 2/3, 2/3)$	-2/9
$(2/3, 1, 0)$	0
$(2/3, 1, 1/3)$	-2/9
$(1, 1, 0)$	0

Table 2: The values of Ω_{FAST} on $(1, 1, 0, w_{ij}, w_{jk}, w_{ki}) \in D_3$.