A Scalable and Unified Framework to Weighted Rank Aggregation

Amir Carmel
Weizmann Institute of Science, Rehovot, Israel

Debarati Das
Pennsylvania State University, University Park, PA, USA

Tien-Long Nguyen
Pennsylvania State University, University Park, PA, USA

Abstract

The rank aggregation problem, seeks to combine multiple rank orderings of the same set of candidates into a single consensus ordering. Such problems arise in diverse domains, including web search, employment, college admissions, and voting. In this work we focus on the 1-median objective: given a set of m rankings over $[n]$, the goal is to compute a ranking that minimizes the sum of its distances to all input rankings.

We study rank aggregation under several classical distance metrics: Ulam distance, Spearman's footrule, Hamming distance, and Kendall-tau, as well as their weighted variants. Our contributions begin with a novel unified framework that identifies a key structural property: it suffices to focus on a small subset of rankings (of size three or five), where the corresponding local one-median provides a good approximation to the global median. This principle extends across these distance measures, yielding a general algorithmic framework for weighted rank aggregation.

Building on this, we present a new approximation algorithm for rank aggregation under the Ulam distance that scales in the Massively Parallel Computation (MPC) model. Our algorithm computes a $(2-\alpha)$-approximation, for a constant $\alpha > 0$, to the 1-median in a constant number of rounds, using local memory sublinear in n (the size of a ranking) and total memory near linear in n.

We further design new MPC approximation algorithms for Spearman's footrule and for the *element-weighted* variants of Hamming and Kendall-tau distances. For each metric, we obtain a $(2-\zeta)$-approximation, for a constant $\zeta > 0$ (which may differ across metrics), to the 1-median in a constant number of rounds, using local memory sublinear in n and total memory linear or near-linear in n.

Moreover, for the Ulam distance, where computing the 1-median is NP-hard [Fischer et al., ESA, 2025], we simplify and strengthen the analysis of Chakraborty et al. [ITCS 2023], obtaining an improved 1.968-approximation that further extends to the weighted setting.

2012 ACM Subject Classification Theory of computation → Massively parallel algorithms; Theory of computation → Facility location and clustering

Keywords and phrases Rank aggregation, 1-median, Ulam distance, Spearman's footrule, Kendall-tau, Hamming distance, weighted metrics, Massively Parallel Computation, Gromov product

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.49

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2605.09653` [10]

Funding *Debarati Das*: This work is supported by NSF grant 2337832.

1 Introduction

Aggregating inconsistent information from diverse sources is a fundamental challenge across disciplines such as social choice theory and information retrieval. The task of reconciling potentially conflicting preferences into a single consensus ranking is known as rank aggregation. This well-studied problem traces back to the late 18th century, when Condorcet and Borda

introduced early voting systems for elections with more than two candidates. In modern settings, rank aggregation continues to play a central role in applications ranging from sports and elections to search engines, databases, web evaluation systems, and statistics [6, 13, 17, 29, 30, 31, 32]. The complexity of the problem is amplified when disagreements or cycles arise in the input rankings, reflecting the conflicts among different sources. Addressing these inconsistencies to produce a meaningful consensus makes rank aggregation a critical and widely impactful data aggregation task.

Given a set of n items, a ranking can be represented by a permutation π on $[n]$. To compare different rankings, several distance measures have been introduced, including Kendall-tau distance [2, 16, 26, 27, 28, 31] (also known as Kemeny distance in the context of rank aggregation), Hamming distance [14], Spearman's footrule distance [16, 17, 31, 38, 37], and Ulam distance [11, 12, 18] (see Section 1.1 for formal definitions). Among the most studied aggregation frameworks are median rank aggregation (or simply rank aggregation) [17, 26, 40, 41] and maximum rank aggregation [5, 8, 35], which aim to find the median and the center ranking of a given set, respectively. In this paper, we consider the continuous version of these problems where the median/center can be any permutation from the input metric space.

Rank aggregation is computationally tractable under Hamming distance and Spearman's footrule, where exact solutions can be obtained using straightforward minimum-cost bipartite matching algorithms. In contrast, under Kendall-tau and Ulam distance, the problem is NP-hard [2, 3, 18]. Nevertheless, a folklore 2-approximation follows directly from the triangle inequality, and a series of works have pushed beyond this trivial approximation. In particular, there exists a PTAS for Kendall-tau [28] while for Ulam distance [11, 12] achieved a 1.999-approximation.

Traditional rank aggregation metrics, as discussed above, treat all errors uniformly, overlooking the fact that in real-world settings, some mistakes are more costly than others. In information retrieval, for instance, misplacing a highly relevant document should incur a much larger penalty than misplacing an irrelevant one. This motivates the use of element weights, where each item is assigned a relevance score, and mistakes involving high-weight elements are penalized more heavily [31, 36].

In many applications, rank aggregation must be performed on massive datasets, where not only is the number of rankings large, but also the size of each ranking. For instance, scholarly search engines such as Corpus rank over 200M papers [34], e-commerce platforms order millions of products [39], and genomic studies combine extensive datasets to prioritize disease-associated genes [1]. Handling such large-scale data naturally motivates the design of efficient parallel algorithms.

In particular, this large-scale setting presents two primary computational challenges: handling a large number of input rankings m, and processing rankings of large size n that may exceed the memory capacity of a single machine. We address the first challenge using sampling techniques, working with only a polylogarithmic-sized sample of the input rankings. The second challenge is more substantial and necessitates distributed parallel computation, naturally motivating the design of efficient parallel algorithms. In particular, when n is large, even storing a single ranking on one machine may be infeasible, requiring rankings to be distributed across multiple machines.

A well-studied framework for this purpose is the Massively Parallel Computation (MPC) model [4, 7, 19, 25]. In this model, each machine has full access to its own local memory, but communication between machines occurs only between rounds. Thus, the round complexity of an algorithm becomes the central performance measure, since network communication

is typically the main bottleneck in practice. The ultimate objective is to design constant-round algorithms, which are highly desirable for large-scale systems. Despite its importance, rank aggregation has not been well studied in the massively parallel setting, and existing algorithms cannot be implemented directly in the MPC model. For instance, algorithms for Hamming or Spearman's footrule rely on min-cost matching, but efficient implementations of matching in constant rounds with memory sublinear in the number of vertices are not known, necessitating new scalable approaches. In this work, we introduce a generalized framework that can be deployed in scalable settings and apply it to several classical rank aggregation measures.

1.1 Problem formulation and preliminaries

Let $(\mathcal{X}, \texttt{dist})$ be an arbitrary metric space. Consider a finite set $P = \{p_1, p_2, \ldots, p_m\} \subseteq \mathcal{X}$. For a given $x \in \mathcal{X}$, we define the cost function as

$$\text{cost}(x, P) = \frac{1}{|P|} \sum_{i=1}^{m} \texttt{dist}(x, p_i).$$

The *1-median problem* seeks to find a point in $\mathcal{X}$ that minimizes this cost function. Let $x^* \in \mathcal{X}$ be an optimal solution for P, that is, $x^* = \arg\min_{x \in \mathcal{X}} \text{cost}(x, P)$, and denote $\text{OPT} = \text{cost}(x^*, P)$ as the optimal cost. Note that x^* may not be unique; when multiple optimal solutions exist, we fix an arbitrary choice.

While our framework is conceptually applicable to general metric spaces, in this paper we focus on the case where $\mathcal{X} = \mathcal{S}_n$ is the set of all permutations on $[n]$ elements, and $\texttt{dist}$ is some distance metric over $\mathcal{S}_n$. This setting is also known as the *rank aggregation problem*. Specifically, we study four classical distance measures on permutations: *Spearman's footrule distance*, *Hamming distance*, *Kendall-tau distance*, and *Ulam distance*, along with the element-weighted variants.

Spearman's footrule distance. measures the ℓ_1 distance between the position vectors of two permutations:

$$\texttt{dist}_F(p, q) = \sum_{i=1}^{n} \Big| p[i] - q[i] \Big|.$$

Hamming distance. measures the number of positions where two permutations differ:

$$\texttt{dist}_H(p, q) = \sum_{i=1}^{n} \mathbb{1}_{p[i] \neq q[i]}.$$

The weighted version penalizes mismatches proportionally to the average weight of the mismatched elements:

$$\texttt{dist}_H^w(p, q) = \sum_{i=1}^{n} \frac{w(p[i]) + w(q[i])}{2} \mathbb{1}_{p[i] \neq q[i]}.$$

Kendall-tau distance. measures the number of elements pairs (e, e') such that the relative order of e and e' is reversed between permutations p and q. Formally,

$$\texttt{dist}_\tau(p, q) = \Big| \big\{ \{e, e'\} : (p^{-1}[e] - p^{-1}[e'])(q^{-1}[e] - q^{-1}[e']) < 0 \big\} \Big|,$$

where $p^{-1}[e]$ is the position of the element e in the permutation p. The weighted version penalizes inversions proportionally to the average weight of the two elements:

$$\texttt{dist}_\tau^w(p,q) = \sum_{\substack{e<e' \\ (p^{-1}[e]-p^{-1}[e'])(q^{-1}[e]-q^{-1}[e'])<0}} \frac{w(e)+w(e')}{2}.$$

Ulam distance. measures the minimum number of move operations required to transform one permutation into another, where a move operation removes an element and reinserts it at a different position. We express this using the Indel distance (also known as LCS distance), which counts single-character insertions and deletions. The Indel distance is exactly twice the Ulam distance, since each move corresponds to one deletion followed by one insertion.

$$\texttt{dist}_U(p,q) = n - LCS(p,q) = \frac{1}{2}\mathrm{ID}\left(p,q\right).$$

In the weighted variant, each insertion or deletion of element i has cost $w(i)$. Thus, the total weighted distance equals the sum of weights of all elements that must be moved to transform one permutation into the other.

Framework. Our framework relies on analyzing the geometric structure of the metric space through the following key quantity. For an element $x \in \mathcal{X}$ and a pair $i, j \in [m]$, we consider the *triangle inequality slack*

$$\Delta_{i,j}(x) = \texttt{dist}(x, p_i) + \texttt{dist}(x, p_j) - \texttt{dist}(p_i, p_j).$$

This quantity is twice the *Gromov product*, a fundamental concept in hyperbolic geometry. By the triangle inequality, we have $\Delta_{i,j}(x) \geq 0$ for all $i, j \in [m]$ and $x \in \mathcal{X}$.

Finally, for any subset $Q \subseteq P$, we define the *total slack* as

$$\Delta(x, Q) = \sum_{\substack{i<j, \\ p_i, p_j \in Q}} \Delta_{i,j}(x). \tag{1}$$

While the total slack depends on the set Q, we omit Q as it is clear from context, and further denote $\Delta^* = \Delta(x^*)$ the total slack of the optimal median. A small total slack of x with respect to Q means that while the points in Q are mutually distant from one another, x serves as an effective median for the entire set. Note that,

$$\Delta(x) = 2\binom{|Q|}{2}\mathrm{cost}(x, Q) - \sum_{\substack{i<j, \\ p_i, p_j \in Q}} \texttt{dist}(p_i, p_j).$$

It follows that an optimal median of Q minimizes the total slack, and in particular

$$\min_{y \in \mathcal{X}} \Delta(y) \leq \Delta^*. \tag{2}$$

The inequality holds because x^*, which is optimal for P, need not be optimal for the subset Q.

MPC Model. Throughout this paper, we assume the input consists of m permutations on $[n]$. In the *MPC model* [4, 7, 19, 25], both the number of machines and the local memory per machine are required to be significantly smaller than the input size. Accordingly, we fix a parameter $0 < \epsilon < 1$, assume each machine has $\widetilde{\mathcal{O}}\left(n^{1-\epsilon}\right)$ memory[1], and aim to minimize the total number of machines used by the algorithm.

[1] Throughout, $\widetilde{\mathcal{O}}(f) = O(f \cdot \mathrm{polylog} f)$

An MPC algorithm proceeds in a sequence of rounds. Within each round, every machine performs computations on the data stored in its memory. No communication is allowed during a round; instead, communication occurs only between rounds, with the restriction that the amount of data received by any machine does not exceed its memory capacity. Furthermore, any data output by a machine must be computed solely from the data already present locally. The input data is initially distributed across the machines.

1.2 Our contribution

In this paper we propose a new unified framework for providing approximate solutions to the 1-median problem for various ranking measures. The key idea can be summarized by the following principle: *If no input point serves as a good median, then there exists a constant-size subset $Q \subseteq P$ such that a good approximation to the 1-median over Q lies within a small distance of the optimal solution for P.* We refer to the approximation over Q as a *local solution* and the optimal median for P as the *global median*.

The main advantage of this framework lies in its efficiency: since the subset Q is small, it is computationally efficient to obtain a local solution y that is optimal or near-optimal for Q, and the distance between x^* and y can be bounded using the slack expressions $\Delta(x^*)$ and $\Delta(y)$.

While Chakraborty, Das, and Krauthgamer [12] also proposed a construction for approximating the median from a small number of input permutations, their approach relies on combinatorial properties specific to the Ulam distance. In contrast, our local-vs-global property is significantly more general, yielding a broader and unified approach that applies to a variety of ranking problems.

Furthermore, the framework of [12] has inherent bottlenecks that limit its applicability in distributed settings. In contrast, our first contribution is to show that our framework is sufficiently robust to be deployed in distributed settings. We design new algorithms in the Massively Parallel Computation (MPC) model that compute approximate medians over such subsets Q. This scalability is crucial for modern applications involving massive datasets, where both the number of rankings and the length n of each ranking are large, making it impractical for a single machine to process even one ranking [1, 34, 39]. Following this we propose the first approximation algorithm breaking the 2-factor barrier for rank aggregation under Ulam distances in the MPC model, using $\widetilde{\mathcal{O}}\left(n^{1-\varepsilon}\right)$ memory per machine, total memory near-linear in n: the size of ranking , and a constant number of rounds.

▶ **Theorem 1.1.** *For any constants $\delta > 0$, $\varepsilon \in (0, 1/8)$,*[2,3] *there exists a polynomial-time MPC algorithm that, given a set of m permutations $P \subseteq \mathcal{S}_n$, computes, with high probability, a $(2 - \alpha + \mathcal{O}(\delta))$-approximation to the 1-median under the Ulam distance, where $\alpha > 0$ is a constant.*[4] *The algorithm uses $\mathcal{O}(1)$ communication rounds, has total space $\widetilde{\mathcal{O}}\left(n^{1+6\varepsilon}\right)$, and requires $\widetilde{\mathcal{O}}\left(n^{1-\varepsilon}\right)$ local memory per machine.*

[2] The algorithm we present in this paper works with $0 < \varepsilon < 1/8$. However, $1/8 < \varepsilon < 1$ is not a barrier to our technique. For a sufficiently small constant $\varepsilon > 0$, the algorithm uses local memory sublinear in n and total memory near-linear in n.

[3] Smaller δ yields better approximations but requires sampling more permutations from the input, increasing both space complexity and running time by constant factors.

[4] Our analysis gives $\alpha = 0.000007$. This constant has not been optimized and can be further improved.

Next, we show the applicability of our framework to weighted rank aggregation by establishing this property for the element-weighted variants of Kendall-tau, Hamming, and Ulam distances, and for Spearman's footrule. More importantly, we show that the framework extends beyond mere generality and remains sufficiently robust even in the weighted setting. Building on this framework, we propose the first approximation algorithms breaking the 2-factor barrier for rank aggregation under Spearman's footrule and the element-weighted variants of Hamming and Kendall-tau distances in the MPC model, using $\widetilde{\mathcal{O}}\left(n^{1-\varepsilon}\right)$ memory per machine, linear or near-linear total memory, and a constant number of rounds.

▶ **Theorem 1.2.** *For any constant $\delta > 0$, and $\epsilon \in (0,1)$, there is a polynomial time MPC algorithm that, given a set of m permutations $P \subseteq \mathcal{S}_n$, with high probability computes the following:*

- $(1.75+\mathcal{O}(\delta))$ *approximation of* 1*-median under Spearman's footrule distance and element-weighted Hamming distance using $\mathcal{O}(1)$ communication rounds, and $\widetilde{\mathcal{O}}(n^{\epsilon})$ processors each with $\mathcal{O}\left(n^{1-\epsilon}\right)$ local memory.*
- $(1.9+\mathcal{O}(\delta))$ *approximation of* 1*-median under element-weighted Kendall-tau distance using $\mathcal{O}(1)$ communication rounds, and $\widetilde{\mathcal{O}}\left(n^{2\epsilon}\right)$ processors each with $\mathcal{O}\left(n^{1-\epsilon}\right)$ local memory.*

Next, we simplify the analysis of [12], improve the approximation factor from 1.999 to 1.968, and extend the result to the weighted Ulam metric. In contrast to prior work, which is restricted to the unweighted setting, we present the first approximation algorithm for computing a median under the element-weighted Ulam distance.

▶ **Theorem 1.3.** *There is a polynomial time algorithm that, given a set of m permutations $P \subseteq \mathcal{S}_n$, provides a 1.968-approximate solution to the* 1*-median problem under the element-weighted Ulam metric with high probability.*

Note that in our problem the total input size is $O(mn)$. A traditional MPC model only enforces that local memory is sublinear in the input size. In contrast, our results guarantee significantly stronger bounds by requiring local memory to be only sublinear in n. This is particularly important since, in most real-world applications, the number of rankings m can be much larger than the size of each ranking n.

Finally, we note that our framework, particularly the local-global principle, is not specific to permutations and may extend to other metric spaces where a below-2 approximation for the 1-median problem remains open. For example, the edit distance metric for the well-known median string problem [11, 21], and the Fréchet distance or Dynamic Time Warping metrics for the average curve problem [9].

1.3 Technical overview

Framework Overview. Our general framework operates under the assumption that no point in P achieves cost better than $(2-\alpha)\mathrm{OPT}$ for some constant $\alpha \in (0,1)$. Otherwise, we can achieve a $(2-\alpha)$-approximation by simply returning the minimum-cost point from the input set. With this assumption, the framework relies on two key properties:

▶ **Property 1** (Universal)**.** *For every $2 \le r \le m$, there exists a subset $Q \subseteq P$ of size r such that*

$$\Delta^* \le \alpha \binom{r}{2} \mathrm{OPT}. \tag{3}$$

▶ **Property 2** (Metric-specific). *There exists a constant r such that for every subset $Q \subseteq P$ of size r and every $y \in \mathcal{X}$,*

$$\texttt{dist}(x^*, y) \leq \Delta(x^*) + \Delta(y). \tag{4}$$

Property 1 holds universally for all metric spaces. We prove this through an averaging argument by considering the pairwise distances of points in P and expressing them using the distances from the optimal median x^*. This intuitively suggests that for any two permutations in Q, their distance is approximately equal to the sum of their respective distances to the optimal median.

However, Property 2 must be established individually for each metric. When it holds and y is an optimal median of Q, Equation (2) implies $\texttt{dist}(x^*, y) \leq 2\Delta^*$. This means x^* and y are close, so y serves as a good median for P, establishing the *local-global property*. For local solutions, where y is not optimal median of Q, we demonstrate how $\Delta(y)$ can be bounded in terms of Δ^* for our specific metrics. A detailed analysis of our framework is given in Section 2.

New Scalable Approximation Algorithms for 1-Median under Ulam Distance. Building on the above framework, in Section 3 we present a new approximation algorithm for the 1-median under the Ulam distance, which we later show can be adapted to the MPC setting. For the Ulam metric, we assume that the set Q contains five permutations. A natural baseline approach for computing the 1-median of the permutations in Q is to apply a dynamic programming algorithm. However, this approach is not space-efficient, as it requires n^5 space, which is prohibitive in the MPC setting.

An alternative is to adapt the strategy from [12]. In this approach, an element is called *bad* if it is not aligned in an optimal alignment between a permutation in Q and the optimal median for at least two permutations in Q. Let the *bad set* denote the collection of all such elements. The algorithm constructs a tournament graph by taking the majority relative order for each pair of elements induced by the permutations in Q, then removes cycles and outputs a topological ordering.

The main bottleneck of this approach is the cycle-removal step. In [12], the algorithm repeatedly identifies the shortest cycle and removes all its vertices. Since the graph is a tournament, the shortest cycle always has length three, and it can be shown that at least one vertex in each such triangle must be bad, which allows bounding the number of good elements removed.

However, this approach does not extend to the MPC model. Because no single machine can store an entire permutation, a 3-cycle (a, b, c) may not be present on any one machine and therefore may go undetected. One possible workaround is to ensure that every triple of elements is collocated on at least one machine. But this leads to further complications: a single element may appear on many machines, and different machines may independently select distinct triangles involving the same vertex (e.g., (a, b, c) and (a, d, e)). This is problematic because the [12] analysis crucially relies on the selected cycles being vertex-disjoint in order to bound the number of good elements removed.

It is nontrivial to reconcile these independently detected cycles while maintaining vertex disjointness in the MPC setting. This difficulty forms the main obstacle to implementing prior approaches. To overcome this, we propose a new framework and algorithm that provides a local solution for the permutations in Q.

First, fix an optimal median of Q, denoted by y. We partition y into blocks of length $\kappa = n^{1-\varepsilon}$, for some constant $\varepsilon > 0$. Our goal is to construct, for each such block of y, a substring of comparable length that provides a good approximation whenever the objective value of the block is not large. We classify an objective value as large if it is at least κ/ρ, where $\rho > 0$ is a small constant to be specified later. If the objective value is large, we instead use a block consisting only of dummy elements, which already yields a sufficiently good approximation. Furthermore, when approximating a block, we focus only on the good elements and ignore the bad elements. This is valid because the total number of bad elements is small and their cumulative contribution to the objective is large, allowing them to be handled arbitrarily. For this we devise a four-step process.

Windows Decomposition. For each permutation $\pi \in Q$, fix an optimal alignment between π and y. Let q be a block of y. Define $e^\ell \in q$ to be the leftmost element of q that is matched in the optimal alignment between π and y, and define $e^r \in q$ to be the rightmost element with this property. Suppose e^ℓ is matched to $\pi[i]$ and e^r is matched to $\pi[j]$. We define π^q to be the substring $\pi[i, j]$. The collection of substrings $\{\pi^q : \pi \in Q\}$ suffices to reconstruct q; we refer to these substrings as the *constructive blocks*. The main challenge is to identify these constructive blocks efficiently.

We now describe a window decomposition for the permutations in Q, beginning with several observations specific to our problem. A key difficulty is that constructive blocks may have highly variable lengths. We first consider the case where at least one constructive block is very large, for example of size greater than $k|q|$ for some sufficiently large constant k. In this situation, the objective value of q is already large, and therefore a trivial approximation using a block of dummy elements suffices.

Next, consider the case where at least two constructive blocks are very small. Then many elements of q do not appear in both of these blocks and hence remain unmatched in at least two permutations from Q. Such elements are therefore *bad elements*. This implies that the number of bad elements in q is large. Since the total number of bad elements across all blocks is small, this scenario cannot occur frequently. Consequently, in this case as well, a good approximation can be obtained by using a trivial block consisting only of dummy elements.

From this point onward, we assume that no constructive block is too large and that at most one constructive block is very small. Even under this assumption, the constructive blocks may still have different sizes. To handle this, we introduce a windowing strategy that, for each permutation $\pi \in Q$, generates windows of variable sizes starting at various indices. We ensure that for every constructive block π^q whose size is neither too large nor too small, there exists a window $\pi[i', j']$ such that

$$|i - i'| + |j - j'| \leq \rho \cdot \mathrm{obj}(q),$$

where $\mathrm{obj}(q)$ denotes the objective value of the block q.

A crucial aspect of this strategy is that we allow windows of size zero to handle very small constructive blocks, and we consider such zero-length windows at multiple starting indices.

Block Reconstruction. Next, using these windows as constructive blocks, we attempt to estimate the blocks of y. Since we do not know in advance which windows correspond to a particular block, we try various choices of five windows. Specifically, we form *constructive groups* by selecting one window from each string in Q using a structured method rather than enumerating all combinations.

Fix an arbitrary choice of five windows (constructive blocks), forming a constructive group. We first construct a graph whose vertices correspond to elements that appear in at least four constructive blocks. This restriction is justified because we only aim to approximate the good elements: any element that does not appear in at least four blocks is classified as a bad element.

For each pair of vertices, corresponding to two elements c and d, we orient the edge between them as follows: we add an edge from c to d if c appears before d in at least three of the five blocks, and an edge from d to c if d appears before c in at least three blocks. If neither direction has a strict majority, we add no edge and delete both vertices c and d. This is valid because the absence of a majority implies that at least one of c or d is a bad element. Since the total number of bad elements is small, the number of good elements deleted in this process is also small.

After this pruning step, the resulting graph is a tournament. This property is crucial for the next phase, where we remove cycles. We iteratively find the shortest cycle in the graph and delete all vertices in that cycle. Because the graph is a tournament, the shortest cycle has length at most three, and among the vertices of such a cycle, at least one must be a bad element. Moreover, since the cycles removed in different iterations are vertex-disjoint, the total number of good elements deleted remains bounded by the number of bad elements.

Once the graph becomes acyclic, we compute a topological ordering of the remaining vertices, which yields a candidate block for the constructive group. For the subsequent dynamic programming step, we ensure that each candidate block has length at least κ. Thus, if the string obtained from the topological ordering has length less than κ, we pad it with dummy elements to reach length κ.

Block Composition via Dynamic Programming. We combine these candidate blocks, each of length at least κ, using dynamic programming to obtain a good approximation of y. Since the dynamic program operates on blocks, the total state space is polynomial in $n/\kappa = n^{\varepsilon}$. Consequently, the DP can be implemented in the MPC setting within a single round by storing all blocks on one machine.

An important subtlety is that, as discussed earlier, for some blocks of y the corresponding constructive blocks may be either very large or very small. In such cases, we approximate the block of y using a block consisting entirely of dummy elements. However, we do not include these dummy-only blocks explicitly in the dynamic program, as doing so would significantly increase the space requirements. Instead, they are handled implicitly within the DP formulation.

The output of the dynamic program is a string that provides a good approximation of y over the good elements. Moreover, each element appears at most once in this string. This property is guaranteed by the construction of the graph used in the previous step, where a vertex corresponding to an element is included only if the element appears in at least four constructive blocks.

Post Processing. The DP output may omit some elements and may include dummy elements As argued earlier, the number of such elements is small and bounded by the number of bad elements. To convert the output into a permutation, we replace each dummy element with a missing element. One simple approach would be to delete all dummy elements and append the missing elements at the end, which would still yield a good approximation. However, our chosen strategy is crucial for enabling an efficient implementation in the MPC setting.

New Scalable Approximation Algorithms for 1-Median with Element Weights. In the element-weighted setting although exact (or almost exact) 1-median algorithms may exist for computing the median of Q, they may not be directly adapted to the massively parallel setting. Such is the case for the rank aggregation problem under Hamming, Spearman's footrule and Kendall-tau. To this end, we design algorithms that are scalable and are well-suited for implementation in the MPC model

Again, the cornerstone of our approach is to devise a consensus strategy among the permutations in Q. We then show that for the relevant r, this consensus is close to the optimal median of the entire set P. We establish these distance bounds using the total slack term, which provides a unified framework across metric spaces. The details and analysis of the algorithms are presented in Section 4.

Hamming Distance. Our algorithm constructs a consensus by selecting the majority element at each position where one exists among the three permutations in Q. The reasoning for selecting the majority as the basis of our consensus strategy is that if the optimal median does not follow the majority, then both input permutations supporting the majority will differ from the optimum, thereby increasing their slack. Since the total slack is assumed to be small (by Equation 3), this situation cannot occur frequently. Positions without majority agreement are filled arbitrarily with the remaining elements. The key insight is that such positions are exactly those contributing to the total slack: since they lack a majority, they affect both the local solution and the global optimum equally. Importantly, this property holds even in the element-weighted setting, making our consensus strategy robust to weights and yielding a 1.75-approximation.

Spearman's Footrule. We construct a consensus by taking the median element at each position across the three permutations, which yields a pseudo-permutation that may contain duplicates. This consensus is then converted into a valid permutation by sorting the elements and reassigning ranks according to their sorted order. While it is clear that taking the median value at each position minimizes the median cost, it is less obvious that this remains optimal once we enforce the consensus to be a valid permutation.

Our key observation is that whenever the optimal median permutation selects a value different from the position-wise median, the slack increases for at least a pair of inputs. For example, suppose at position i the three input permutations π_1, π_2, π_3 have values $a \geq b \geq c$. If the median instead uses some $b' \neq b$ (say $b' > b$) at index i, then the slack of π_2 and π_3 at this position increases by $2(b' - b)$. Since the total slack is assumed to be small, such deviations cannot occur frequently. Thus, even under the constraint that the output must be a permutation, taking the position-wise median remains the best consensus strategy. This transformation ultimately yields the closest valid permutation to our consensus under the Spearman's footrule metric, resulting in an overall 1.75-approximation.

Kendall-tau Distance. The Kendall-tau metric compares inversions between pairs of elements. The consensus is determined by majority vote over element pairs rather than positions. We construct a tournament graph where vertex a precedes vertex b if a appears before b in at least two of the three permutations. The resulting feedback arc set problem, solved via the KWIK-SORT algorithm, yields a permutation that respects the majority preference on most pairs. While the KWIK-SORT algorithm was applied to rank aggregation under the Kendall-tau metric in [2] to obtain an approximate 1-median solution, the weighted setting requires a different approach. Using our framework, a weighted tournament is derived from the three input permutations, with edge weights assigned so that the triangle inequality holds, thereby enabling the use of the KWIK-SORT algorithm.

Improved Approximation for Ulam Distance. Unlike the previous metrics, consensus under Ulam distance requires subsets Q of size 5. This follows from a more general property: for any two permutations x, y, we show the distance from x to y is at most $\Delta(x) + \Delta(y)$. Combined with the universal Property 1, this enables us to improve the metric-specific analysis of [12] and naturally extend the result to the element-weighted variant.

MPC Algorithm for Approximating 1-Median. We start by describing how permutations are represented in the MPC model and how pairwise distances are computed, primitives common to all metrics.

Storing and Computing Distance in MPC. A key challenge in the MPC model is that each machine has only $\widetilde{\mathcal{O}}(n^{1-\varepsilon})$ local memory, which is insufficient to store an entire permutation; consequently, each permutation must be distributed across at least $\widetilde{\mathcal{O}}(n^{\varepsilon})$ machines. This sublinear memory constraint makes it nontrivial to compute a local solution and validate its cost even for a small subset Q of permutations, necessitating specialized MPC algorithms tailored to different distance metrics. Hamming and Spearman's footrule distances can be computed in constant rounds using $\mathcal{O}(n)$ total space via block-wise aggregation. Kendall-tau distance requires $n^{2\varepsilon}$ machines to handle inversions between all block pairs, resulting in a total of $\mathcal{O}\left(n^{1+\varepsilon}\right)$ space. Computing Ulam distance in MPC is more challenging, requiring the $(1+\lambda)$-approximation algorithm of [20] with $\widetilde{\mathcal{O}}\left(n^{1+\varepsilon}\right)$ space.

A Sublinear Total Space Framework. We address the constraint on total memory through a sampling-based approach. Rather than enumerating all possible candidates, which may be as many as $\mathcal{O}(m^r)$, we show that it suffices to consider only $\mathcal{O}(\log(n))$ randomly selected input permutations and $\mathcal{O}(\log(n))$ local solutions derived from $\mathcal{O}(\log(n))$ random subsets of size r. To identify the best candidates, instead of computing their exact costs, we leverage results from [23, 24], which demonstrate that, with high probability, a $(1+\delta)$-approximate solution to the optimal candidate can be found by selecting the permutation with the minimum cost over a random sample of $\mathcal{O}\left(\log(n)/\delta^2\right)$ input permutations.

Next, we outline how to compute a local solution of the set Q under Ulam, Hamming, Spearman's footrule, and Kendall-tau distances using constantly many communication rounds.

Ulam distance. To employ our new algorithm for the Ulam distance in the MPC setting, we proceed in three phases.

First, we distribute the task of constructing candidate blocks. For each group of five windows, we assign a specific machine to build the tournament graph, remove cycles, and output the resulting block. These results must then be aggregated to find the optimal combination of blocks. A direct approach would be to collect all candidate blocks onto a single machine to run the dynamic programming. However, this is not space-efficient, as the total size of these blocks may exceed the local memory of a single machine. To overcome this, we observe that the dynamic program does not need the actual strings; it only requires their objective values, lengths, and the starting and ending indices of the corresponding windows. This summary information is small enough to fit in the memory of one machine, allowing us to compute the optimal sequence of blocks efficiently.

The solution to the dynamic program implicitly defines a string consisting of n^{ε} blocks. Each block is either a candidate block stored on one of the distributed machines or a logical block consisting entirely of dummy elements. By backtracking through the solution, we can identify the specific machines holding the chosen candidate blocks and assign new machines to generate the required blocks of dummy elements.

It remains to transform this distributed string into a valid permutation of length n. We first truncate the string by removing the leftmost dummy elements until the total length is exactly n. This is a simple counting task that can be performed using a broadcast tree of constant depth. Next, we must replace each remaining dummy element with a unique unused element. Our strategy is to pair the distinct unused elements with the remaining dummy positions according to their sorted order. That is, the k-th smallest unused element should be placed in the k-th available dummy position.

To implement this matching, we need to compute the global rank of each unused element and each dummy position. Since the data is distributed, each machine first counts the unused elements it holds locally. Then, using a broadcast tree, each machine computes the total count of unused elements residing on all preceding machines. This allows every machine to determine the exact global rank of its elements. We apply the same procedure to rank the dummy positions. Finally, the elements and positions with matching ranks are routed to the same machine to complete the permutation.

Hamming distance. In the algorithm for Hamming distance, (i) a majority element is assigned to each position if it exists among the permutations in Q. (ii) Positions without majority agreement are filled arbitrarily with remaining elements. The first step can be simulated in MPC similar to computing the Hamming distance: we bring the information from the i'th block of each permutation in Q to a single machine and decide the majority.

The difficulty lies in the second step, as the list of unused elements and unassigned positions can be large and may not fit in a single machine. A simple approach is to assign unused elements in each i block to unassigned positions in the same block. However, this may not be feasible as the number of unused elements and unassigned positions in a block may differ. To resolve this, we consider the sorted list of unused elements and unassigned positions, and aim to pair them according to their order in the sorted lists. This requires computing the rank of each unused element relative to all unused elements, and the rank of each unassigned position relative to all unassigned positions. To implement this, we let each i'th machine keep the unused elements in the i'th block. The rank of each element can be computed if each machine knows the number of unused elements in all the previous blocks. This information can be made available to all machines by using a broadcast tree of depth $\max(1, \frac{2\varepsilon-1}{1-\varepsilon})$. We proceed similarly for unassigned positions. Finally, the elements and positions with the same rank are sent to the same machine.

Spearman's Footrule. In the algorithm for Spearman's footrule distance, we first compute a pseudo-permutation by assigning the median element at each position across the permutations in Q. The output permutation is obtained by sorting the elements in the pseudo-permutation and reassigning ranks according to their sorted order. The first step can be implemented in MPC similar to Hamming distance. The second step pose a similar challenge as in Hamming distance, since the elements are distributed across multiple machines. By sorting all elements in the pseudo-permutation, we reduce this problem to the pairing problem as in Hamming distance.

Kendall-tau distance. For Kendall-tau distance, recall that we need to apply the KWIK-SORT algorithm to solve the feedback arc set problem on the majority graph induced by Q. The KWIK-SORT algorithm starts with a random vertex as pivot, and partitions the remaining vertices into two sets according to the direction of the edges between them and the pivot. We then recurse on the two sets, and concatenate the results. We follow a constant

rounds MPC implementation of KWIK-SORT presented in [22]. The approach to simulate this in constant MPC rounds is to pick multiple pivots in each round. These pivots form a decision tree, which partitions the vertices into multiple sets. We then recurse on each set in parallel. However, if we directly follow the algorithm in [22], we will violate the local space constraint. In particular, [22] allows the local space to be $\widetilde{\mathcal{O}}(n)$, hence, they can choose $\mathcal{O}\left(n^{1/2}\right)$ pivots simultaneously in each round. Our local space may not be sufficient to accommodate this many vertices. We overcome this by choosing only $\mathcal{O}\left(n^{1-\varepsilon}\right)$ vertices as pivots. Due to the fact that the size of partitioned sets can not be too large [22, Lemma 7], as long as the number of pivots is $\Omega(\log(n))$, the number of rounds remains constant. Another challenge is that in [22], the total space is $\mathcal{O}\left(n^2\right)$, as they need to store the edges explicitly. We resolve this by observing that we can access the direction of an edge by accessing the positions of two elements in the permutations in Q. This allows us to store the edges implicitly, keeping the total space at $\mathcal{O}(n)$.

1.4 Organization

The remainder of the paper is organized as follows. Section 2 formalizes the general framework: we prove the universal Property 1 (Lemma 2.1) and combine it with the metric-specific Property 2 to obtain Algorithm 1 and its approximation guarantee (Theorem 2.2). Section 3 presents our new approximation algorithm `ScalableMedianReconstruct` for rank aggregation under the Ulam metric, stating its main guarantee (Lemma 3.1) and giving a high-level overview of its four steps; the full algorithm description and analysis are deferred to the full version [10]. Section 4 establishes the metric-specific Property 2 for Hamming, Spearman's footrule, Kendall-tau, and Ulam distances, yielding our offline approximation results for the weighted setting. The detailed proofs from Section 4, the MPC implementations of all four local-solution algorithms, together with the proofs of our main MPC results (Theorems 1.1 and 1.2), are also given in the full version [10].

2 General Framework Analysis

In this section, we establish our framework and prove the components that are universal across all metric spaces. We begin by formalizing Property 1 in the following lemma.

▶ **Lemma 2.1.** *Let P be a finite set of points in a metric space $(\mathcal{X}, \texttt{dist})$. Assume that the number of points $p \in P$ for which $\mathrm{cost}(p, P) \le (2-\alpha)\mathrm{OPT}$ is at most δm, for $\delta \in [0, 1-\alpha]$. Let Q be a random subset of P of size $r \ge 2$. Then, $\mathbb{E}_Q[\Delta^*] \le \binom{r}{2}(\alpha + 2\delta)\mathrm{OPT}$.*

Proof. Let $F = \{p \in P \mid \mathrm{cost}(p, P) \le (2-\alpha)\mathrm{OPT}\}$ be the set of points from P with good approximation factor. From our assumption on the size of F, we have $|F| \le \delta m$, so we can lower bound the total cost

$$\sum_{p\in P} \mathrm{cost}(p,P) = \sum_{p\in F} \mathrm{cost}(p,P) + \sum_{p\in P\setminus F} \mathrm{cost}(p,P) \ge \sum_{p\in P\setminus F} \mathrm{cost}(p,P) \ge (2-\alpha)(1-\delta)m\mathrm{OPT}.$$

On the other hand, we can express the same total cost by summing over all pairwise distances,

$$\sum_{p\in P} \mathrm{cost}(p,P) = \sum_{p\in P} \frac{1}{m}\sum_{i=1}^{m} \texttt{dist}(p,p_i) = \frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m} \texttt{dist}(p_i,p_j).$$

Using the definition of triangle inequality slack, we can write,

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m}\texttt{dist}(p_i,p_j) &= \frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m}\left(\texttt{dist}(x^*,p_i)+\texttt{dist}(x^*,p_j)-\Delta^*_{i,j}\right)\\ &= \frac{2}{m}\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m}\texttt{dist}(x^*,p_i)-\frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m}\Delta^*_{i,j}\\ &= 2(m-1)\mathrm{OPT}-\frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m}\Delta^*_{i,j}\end{aligned}$$

Combining both bounds,

$$(2-\alpha)(1-\delta)m\mathrm{OPT}\leq 2(m-1)\mathrm{OPT}-\frac{1}{m}\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m}\Delta^*_{i,j}.$$

Rearranging,

$$\sum_{i=1}^{m}\sum_{\substack{j=1,\\ j\neq i}}^{m}\Delta^*_{i,j}\leq(m\alpha(1-\delta)+2m\delta-2)m\mathrm{OPT}\leq(m\alpha+2m\delta-2)m\mathrm{OPT}.$$

Since each pair $\{i,j\}$ with $i\neq j$ appears twice in the double sum, and $\Delta^*_{i,i}=0$,

$$\sum_{1\leq i<j\leq m}\Delta^*_{i,j}\leq\frac{1}{2}(m\alpha+2m\delta-2)m\mathrm{OPT}.$$

For a subset Q of size r drawn uniformly at random from P, using linearity of expectation,

$$\mathbb{E}_Q[\Delta^*]=\mathbb{E}_Q\left[\sum_{\substack{i<j\\ p_i,p_j\in Q}}\Delta^*_{i,j}\right]=\sum_{1\leq i<j\leq m}\Delta^*_{i,j}\cdot\Pr[p_i\in Q,p_j\in Q].$$

The probability that both p_i and p_j are selected in a random subset of size r is

$$\Pr[p_i\in Q,p_j\in Q]=\frac{\binom{m-2}{r-2}}{\binom{m}{r}}.$$

Therefore,

$$\begin{aligned}\mathbb{E}_Q[\Delta^*] &= \frac{\binom{m-2}{r-2}}{\binom{m}{r}}\sum_{1\leq i<j\leq m}\Delta^*_{i,j}\leq\frac{\binom{m-2}{r-2}}{\binom{m}{r}}\cdot\frac{1}{2}(m\alpha+2m\delta-2)m\mathrm{OPT}\\ &= \frac{r(r-1)}{2(m-1)}\cdot(m\alpha+2m\delta-2)\mathrm{OPT}\leq\binom{r}{2}(\alpha+2\delta)\mathrm{OPT}\end{aligned}$$

where the last inequality holds for $\delta\leq 1-\alpha$. ◀

We formalize Property 2 as we apply it in our analysis.

▶ **Property 2** (Metric-specific). *For our framework to apply to a metric space $(\mathcal{X},\texttt{dist})$, we require the existence of constants $C>0$ and $r\geq 2$, and an algorithm that, given any subset $Q\subseteq P$ of size r, produces a solution y such that $\texttt{dist}(x^*,y)\leq C\cdot\Delta^*$.*

The challenge is to design such an algorithm and establish this property for each specific metric space.

We now combine both properties to design a general framework for the 1-median problem in the sublinear regime. Our framework operates by establishing that if only few points in P achieve a $(2-\alpha)$-approximation for some small constant α, then there exist many constant-size subsets $Q \subseteq P$ with small total slack. Following Property 2 the algorithm applied on such Q produce a point that is close to the optimal global median.

Hence, we construct a set of candidates $\mathcal{C}$ such that with high probability one of the candidates achieves a $(2-\alpha)$-approximation. We employ a sublinear approach that was developed by Indyk [23, 24] for the discrete 1-median problem, where the medians are restricted to a specific set of points. In [24] it was shown that sampling $\mathcal{O}\left(\delta^{-2}\log n\right)$ points is sufficient to identify a candidate that achieves a $(1+\delta)$-approximation to the best median in the candidate set $\mathcal{C}$ with high probability. See Algorithm 1 for the complete pseudocode.

Algorithm 1 A space efficient general framework for 1-median approximation.

Input : Set $P \subseteq \mathcal{X}$, constant r, $\delta > 0$
Output : Approximate 1-median solution
1 $\mathcal{C} \leftarrow$ a set of $\mathcal{O}(\log n/\delta)$ points sampled uniformly at random from P ;
2 **for** $i \in \{1, 2, \ldots, \mathcal{O}(\log n/\delta)\}$ **do**
3 $\quad Q \leftarrow$ a set of r permutations sampled uniformly at random from P;
4 $\quad y \leftarrow$ `localAlg`(Q); `// compute a local solution over` Q
5 $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{y\}$;
6 Sample a set S of $\mathcal{O}\left(\log n/\delta^2\right)$ permutations uniformly at random from P ;
7 **return** $x := \arg\min_{x\in\mathcal{C}} \mathrm{cost}(x, S)$;

▶ **Theorem 2.2.** *Let $t(n)$ denote the time required to compute the distance between two permutations of length n. Assume Property 2 holds with constants C and r, and there is an algorithm that, given any subset $Q \subseteq P$ of size r, in time $t_\ell(n)$ produces a solution y such that $\mathtt{dist}(x^*, y) \le C \cdot \Delta^*$. Then Algorithm 1 returns with high probability a $(2 - \frac{1}{C\binom{r}{2}+1} + \mathcal{O}(\delta))$-approximation to the 1-median problem on P using $\widetilde{\mathcal{O}}(t_\ell(n) + t(n))$ time.*

Proof.

Approximation ratio analysis. Let $\alpha = (C\binom{r}{2}+1)^{-1}$. If the number of points $p \in P$ for which $\mathrm{cost}(p, P) \le (2-\alpha)\mathrm{OPT}$ is at least δm, then w.h.p such a point is added to the set of candidates $\mathcal{C}$. Otherwise, by Lemma 2.1, expected total slack of Q for every Q sampled is at most $\binom{r}{2}(\alpha + 2\delta)\mathrm{OPT}$. Using Markov inequality, in Line 1, with high probability, we sample a set Q with at most $\binom{r}{2}(\alpha + 3\delta)\mathrm{OPT}$. The output y of `localAlg`(Q) is added to $\mathcal{C}$, furthermore, following Property 2, $\mathtt{dist}(x^*, y) \le C\Delta^* \le C\binom{r}{2}(\alpha + 3\delta)\mathrm{OPT}$. Then, by the triangle inequality,

$$\begin{aligned}\mathrm{cost}(y, P) \le \frac{1}{m}\sum_{p\in P}(\mathtt{dist}(y, x^*) + \mathtt{dist}(x^*, p)) &\le C\binom{r}{2}(\alpha + 3\delta)\mathrm{OPT} + \mathrm{OPT} \\ &\le (C\binom{r}{2}(\alpha + 3\delta) + 1)\mathrm{OPT}.\end{aligned}$$

Using Indyk's sampling method and our selection of α, we obtain Algorithm 1 return a $(2 - \alpha + \mathcal{O}(\delta))$-approximation.

Running time analysis. Line 1 repeats $\mathcal{O}(\log(n)/\delta)$ times, each executing `localAlg` in $t_\ell(n)$ time. Line 1 computes the cost of $\mathcal{O}(\log(n)/\delta)$ candidates to set S of size $\mathcal{O}\left(\log(n)/\delta^2\right)$ in $\mathcal{O}\left(\log(n)^2 t(n)/\delta^3\right)$ time. For a constant δ, the total running time is then $\widetilde{\mathcal{O}}(t_\ell(n) + t(n))$. ◀

3 New Approximation Algorithm for Rank Aggregation under Ulam Metric

In this section, we present our new approximation rank aggregation algorithm under Ulam metric. Using the framework we developed in Algorithm 1, it suffices to design a scalable algorithm on input Q consisting of r permutations, producing a permutation y_{offline} that is close to x^*.

Our contribution is a new offline polynomial-time algorithm, `ScalableMedianReconstruct`, which we implement in the MPC model using $\mathcal{O}(1)$ rounds.

This algorithm satisfies a variant of Property 2, as stated in Lemma 3.1. The detailed algorithm description, its analysis, and the MPC implementation are deferred to the full version [10].

▶ **Lemma 3.1.** *Let $Q = \{\pi_1, \pi_2, \ldots, \pi_5\}$ be a set of five permutations from P. The output y_{offline} of* `ScalableMedianReconstruct` *with input Q satisfies*

$$\mathrm{ID}(y_{\text{offline}}, x^*) \le 0.0009 \sum_{i=1}^{5} \mathrm{ID}(x^*, \pi_i) + 266\Delta(x^*). \tag{5}$$

Additionally to Lemma 3.1, we require the following structural lemma, which extends Lemma 2.1 with additional properties that we exploit in the Ulam case.

▶ **Lemma 3.2.** *Let $\alpha \le 1$ and $\delta \le \frac{\alpha}{100}$. Let P be a finite set of m points in a metric space $(\mathcal{X}, \texttt{dist})$, and assume that $|\{p \in P : \mathrm{cost}(p, P) \le (2-\alpha)\mathrm{OPT}\}| \le \delta m$. Let Q be a uniformly random subset of P of size 5. Then with probability at least 0.008, both of the following hold:*
(P1) $\texttt{dist}(q, x^*) \le (1+\alpha)\mathrm{OPT}$ *for all $q \in Q$, and*
(P2) $\Delta(x^*) \le 510\alpha \cdot \mathrm{OPT}$.

Proof. Let

$$F_0 = \{p \in P \mid \mathrm{cost}(p, P) < (2-\alpha)\mathrm{OPT}\},$$
$$F_1 = \{p \in P \mid (1-\alpha)\mathrm{OPT} \le \texttt{dist}(x^*, p) \le (1+\alpha)\mathrm{OPT}\}.$$

Recall $|F_0| = \delta m$ and further denote $|F_1| = \gamma m$ for some $\gamma \in [0, 1]$. Then,

$$\begin{aligned}
\mathrm{OPT} &= \frac{1}{m} \sum_{p \in P} \texttt{dist}(x^*, p) \\
&\ge \frac{1}{m} \sum_{p \in F_1} \texttt{dist}(x^*, p) + \frac{1}{m} \sum_{p \in P \setminus (F_0 \cup F_1)} \texttt{dist}(x^*, p) \\
&\ge \gamma(1-\alpha)\mathrm{OPT} + (1-\gamma-\delta)(1+\alpha)\mathrm{OPT}.
\end{aligned}$$

Solving for γ, we obtain $\gamma \ge \frac{1}{2} - \frac{(1+\alpha)\delta}{2\alpha} \ge \frac{1}{2} - \frac{1}{100}$, where the last inequality follows from our assumptions on α and δ.

Consider a random subset Q of five permutations chosen uniformly from P. Let B denote the bad event that Q does not satisfy both $\Delta(x^*) \le 500(\alpha + 2\delta)\mathrm{OPT}$ and $Q \subseteq F_1$.

For the first condition, by Lemma 2.1 and Markov's inequality,

$$\mathbb{P}[\Delta(x^*) > 500(\alpha + 2\delta)\mathrm{OPT}] \leq \tfrac{1}{50}.$$

Since $\delta \leq \frac{\alpha}{100}$, we have $500(\alpha + 2\delta) \leq 510\alpha$, and therefore $\mathbb{P}[\Delta(x^*) > 510\alpha \cdot \mathrm{OPT}] \leq \frac{1}{50}$.

For the second condition, the probability that not all five sampled points are from F_1 is $1 - \gamma^5 \leq 1 - \left(\frac{1}{2} - \frac{1}{100}\right)^5 < 0.972$. By the union bound, $\mathbb{P}[B] \leq \frac{1}{50} + 0.972 = 0.992$. ◀

▶ **Theorem 3.3.** *For any constant $\delta > 0$, there is a polynomial time algorithm that, given a set of m permutations $P \subseteq \mathcal{S}_n$, with high probability computes a $(2-\alpha+\mathcal{O}(\delta))$ approximation of 1-median under Ulam distance, where $\alpha > 0$ is a constant.*

Proof of Theorem 3.3. We utilize the framework of Algorithm 1 with $r = 5$ and in place of `localAlg` in Line 1, we use our algorithm `ScalableMedianReconstruct`. Our proof relies on Lemma 3.2, which describes a structural property of the input set under Ulam distance.

Fix $\alpha = 0.000007$. If the number of permutations in $p \in P$ for which $\mathrm{cost}(p, P) \leq (2 - \alpha)\mathrm{OPT}$ is at least δm, then w.h.p, such a permutation is included in $\mathcal{C}$ in Line 1 of Algorithm 1.

Otherwise, Lemma 3.2 implies that a random set Q of 5 permutations satisfies properties (P 1) and (P 2) with probability at least 0.008. As we sample $\mathcal{O}(\log n)$ sets in Line 1, w.h.p., such a set Q is included in the samples.

In this case, the candidate set $\mathcal{C}$ contains a permutation

$$y = \texttt{ScalableMedianReconstruct}(\pi_1, \ldots, \pi_5).$$

From Lemma 3.1 and our choice of α, we have:

$$\mathrm{ID}(y, x^*) \leq 0.009(1 + \alpha)\mathrm{OPT} + 266 * 510\alpha\mathrm{OPT} \leq 0.96\mathrm{OPT}.$$

Consequently, by the triangle inequality:

$$\mathrm{cost}(y, P) \leq \frac{1}{m}\left(\sum_{i=1}^{m} (\mathrm{ID}(y, x^*) + \mathrm{ID}(x^*, p_i))\right) \leq \mathrm{ID}(y, x^*) + \mathrm{OPT} \leq 1.96\mathrm{OPT}.$$

In all cases, the candidate set $\mathcal{C}$ contains a permutation achieving a $(2 - \alpha)$-approximation to the optimal 1-median. Choosing a candidate from $\mathcal{C}$ using Indyk's sampling method [24] (lines 6–7 of Algorithm 1) returns a permutation with cost at most $(2 - \alpha + \mathcal{O}(\delta))\mathrm{OPT}$.

As the algorithm `ScalableMedianReconstruct` runs in time polynomial in n, it is clear that the overall algorithm runs in polynomial time. ◀

Overview of ScalableMedianReconstruct Algorithm

Our objective is to compute a good estimation of the median of Q, defined as

$$y^* = \arg\min_{z \in \mathcal{S}_n} \sum_{i=1}^{5} \mathrm{ID}(\pi_i, z).$$

To outline the algorithm, we first fix an optimal alignment between y^* and each input permutation π_i for $i = 1, \ldots, 5$. Consider a partition of y^* into n^ε disjoint contiguous blocks $y^*[\ell_j, r_j)$ of size $\kappa = n^{1-\varepsilon}$. The fixed alignments induce a corresponding decomposition of

each π_i into substrings $\pi_i[\alpha_{i,j}, \beta_{i,j})$, such that the block $y^*[\ell_j, r_j)$ aligns with $\pi_i[\alpha_{i,j}, \beta_{i,j})$. We refer to each $\pi_i[\alpha_i, \beta_{i,j})$ as a *constructive block*, and to $\{\pi_i[\alpha_{i,j}, \beta_{i,j})\}_{i=1}^5$ as a *constructive group*. Each block $y^*[\ell_j, r_j)$ can be approximately reconstructed using the corresponding constructive group $\{\pi_i[\alpha_{i,j}, \beta_{i,j})\}_{i=1}^5$. We refer to these approximations as *candidate blocks*.

The challenge lies in efficiently identifying these constructive blocks and combining the candidate blocks to obtain an overall approximation of y^*. The `ScalableMmedianReconstruct` algorithm accomplishes this through the following four steps:

- `Windows Decomposition`: Each string $\pi_i \in Q$ is decomposed into windows, which serve as *constructive blocks* for building the blocks of y^*. We then form *constructive groups* by selecting one window from each string using a structured method, rather than enumerating all combinations.
- `Block Reconstruction`: For each constructive group, a *candidate block* is computed using the `Block Reconstruction` algorithm. These candidate blocks serve as primitives for the next step.
- `Block Decomposition via Dynamic Programming`: The candidate blocks from the previous step are combined to form an *intermediate string* s via a dynamic programming approach. The intermediate string s consists of n^{ε} blocks, each being either a string created in the `Block Reconstruction` step or a string of dummy elements of length κ.
- `PostProcessing`: The intermediate string s has length at least n and contains no repeated elements, but it may not be a permutation due to missing elements and the presence of dummy elements. To transform s into a permutation y_{offline}, we replace the dummy elements with the unused elements.

4 New Approximation Algorithms for Weighted Rank Aggregation

In this section we prove Property 2 for each metric space. We show that for Hamming, Kendall-tau and Spearman's footrule, $r = 3$, and for the Ulam metric $r = 5$.

Using Theorem 2.2 we then obtain the following result.

▶ **Theorem 4.1.** *Algorithm 1 provides a:*

- $(1.75 + O(\delta))$*-approximation for the 1-median problem in* $(\mathcal{S}_n, \mathtt{dist}_H^w)$ *and* $(\mathcal{S}_n, \mathtt{dist}_F)$ *in linear time.*
- $(1.9 + O(\delta))$*-approximation for the 1-median problem in* $(\mathcal{S}_n, \mathtt{dist}_\tau^w)$ *in* $\mathcal{O}(n \log n)$ *time.*
- $(1.9677 + O(\delta))$*-approximation for the 1-median problem in* $(\mathcal{S}_n, \mathtt{dist}_U^w)$ *in* $\mathcal{O}\left(n^{17}\right)$.

Ulam and Kendall-tau analysis. The analysis for Ulam and Kendall-tau metrics requires additional notation to handle unaligned elements and pairs. For any $x, p \in \mathcal{S}_n$, we denote by I_p^x the set of unaligned characters of x with p. For the Ulam metric, these are unaligned characters of x with respect to some alignment between p and x, while in the Kendall-tau case, I_p^x represents the set of unaligned pairs.

We also establish a relationship between $I_p^x \cap I_q^x$ and the respective distances, in $\mathtt{dist}_\tau^w$ and $\mathtt{dist}_U^w$. For any set S of elements (characters or pairs), we use $\|S\|$ to denote the sum of weights of elements in S.

▶ **Proposition 4.2.** *For every* $p, q, x \in \mathcal{S}_n$,

$$\|I_p^x \cap I_q^x\| \leq \mathtt{dist}(p, x) + \mathtt{dist}(q, x) - \mathtt{dist}(p, q),$$

Where $\mathtt{dist}$ *is either* $\mathtt{dist}_\tau^w$ *or* $\mathtt{dist}_U^w$.

4.1 Hamming

For the weighted Hamming distance, we design a majority-based algorithm for subsets Q of size 3. Algorithm 2 constructs a consensus permutation by assigning the majority element to each position where one exists among the three input permutations. Positions without majority agreement are filled arbitrarily using the remaining unassigned elements.

Algorithm 2 Majority-based median for weighted Hamming.

Input : Q of size 3
Output : Local solution $y \in \mathcal{S}_n$

Initialize $U \leftarrow [n]$ `// Set of unassigned elements`
for *each position* $k \in [n]$ **do**
  if *there exists element* e *appearing at position* k *in at least 2 permutations in* Q **then**
    Set $y[k] \leftarrow e$
    Remove e from U
  else
    Mark position k as unresolved
for *each unresolved position* k *(in arbitrary order)* **do**
  Assign $y[k]$ to any element from U
  Remove the assigned element from U
return y

Algorithm 2 operates in linear time and space complexity. To verify correctness, observe that its output y constitutes a valid permutation since no element can simultaneously be a majority in two distinct positions. We now proceed to demonstrate that the weighted Hamming distance satisfies Property 2.

▶ **Lemma 4.3.** *For the metric space* $(\mathcal{S}_n, \texttt{dist}_H^w)$*, any subset* $Q \subseteq \mathcal{S}_n$ *of size* 3*, and any local solution* y *obtained using Algorithm 2, we have* $\texttt{dist}_H^w(x^*, y) \leq \Delta^*$.

4.2 Spearman's footrule

For the Spearman's footrule distance, we design a position-wise median algorithm for subsets Q of size 3. Algorithm 3 operates in two phases: first, it computes a pseudo-permutation z by taking the median element at each position independently among the three input permutations, second, since z may not constitute a valid permutation due to potential duplicate elements, it converts z to a valid permutation by sorting the element-position pairs and reassigning ranks according to the sorted order of elements.

Algorithm 3 operates in linear time and space complexity as the sorting step can be implemented using counting sort. Observe that the output y is a valid permutation as the sorting step ensures each element from $[n]$ appears exactly once. The algorithm minimizes position-wise Spearman's footrule distances in the first phase, then finds the closest valid permutation in the second phase.

We use the following simple rearrangement inequality (a special case of more general results in Day [15]) to establish our results.

▶ **Lemma 4.4** (Day [15])**.** *If* $a_1 \leq a_2 \leq \ldots \leq a_n$ *and* $b_1 \leq b_2 \leq \ldots \leq b_n$*, then,*

$$\sum_{k=1}^{n} |a_k - b_k| = \min_{\pi \in \mathcal{S}_n} \sum_{k=1}^{n} |a_k - b_{\pi(k)}|.$$

Algorithm 3 Position-wise median for Spearman's footrule.

Input : Q of size 3
Output : Local solution $y \in \mathcal{S}_n$
Initialize pseudo-permutation z of length n
for *each position* $k \in [n]$ **do**
  $z[k] \leftarrow \texttt{median}(Q, k)$ `// median element at position` k
`// Convert pseudo-permutation to valid permutation`
Let $S = \{(z[k], k) : k \in [n]\}$ `// Element-position pairs`
Sort S by element value to get $S' = \{(e_1, k_1), (e_2, k_2), \ldots, (e_n, k_n)\}$
for $i = 1$ *to* n **do**
  $y[k_i] \leftarrow i$ `// Assign rank` i `to position` k_i
return y

From Lemma 4.4, y is the closest permutation to the pseudo-permutation z.

$$\texttt{dist}_F(z, y) = \min_{x \in \mathcal{S}_n} \texttt{dist}_F(z, x) \tag{6}$$

To show that the Spearman's footrule distance satisfies Property 2, we bound the distance from any permutation to z.

▶ **Lemma 4.5.** *For every* $x \in \mathcal{S}_n$, $\texttt{dist}_F(x, z) \leq \frac{1}{2}\Delta(x)$.

Combining Lemma 4.5 and Equation 6, with the triangle inequality, we have

$$\texttt{dist}_F(x^*, y) \leq \texttt{dist}_F(x^*, z) + \texttt{dist}_F(z, y) \leq \frac{1}{2}\Delta^* + \frac{1}{2}\Delta^* = \Delta^*.$$

4.3 Kendall-tau

For the weighted Kendall-tau distance, we propose an algorithm that constructs a tournament graph and solves the resulting feedback arc set problem. The algorithm builds a majority graph G_Q with vertex set $[n]$, where a directed edge (a, b) exists if element a precedes element b in at least two of the three input permutations from Q. Each edge (a, b) is assigned weight $\frac{1}{2}(w(a) + w(b))$. To solve the feedback arc set problem on G_Q, we employ the KWIK-SORT algorithm, which provides a 2-approximation guarantee when edge weights satisfy the triangle inequality [2]. This algorithm runs in $\mathcal{O}(n \log n)$ time and uses linear space [22].

For $x \in \mathcal{S}_n$, define B_x as the set of element pairs that are unaligned when comparing x with at least two permutations from $Q = \{\pi_1, \pi_2, \pi_3\}$,

$$B_x = \bigcup_{1 \leq i < j \leq 3} (I^x_{\pi_i} \cap I^x_{\pi_j}).$$

Note that for a pair of elements $\{a, b\}$ where a precedes b in permutation x, we have $\{a, b\} \notin B_x$ if and only if a precedes b in the majority of permutations from Q. Also, using Proposition 4.2, we have $\|B_{x^*}\| \leq \Delta^*$, where $\|B_{x^*}\|$ denotes the sum of weights of elements in B_{x^*}.

Since x is a valid permutation, we can transform G_Q into an acyclic graph by redirecting edges of total weight at most $\|B_x\|$. Moreover, every orientation of the edges that transforms G_Q into an acyclic graph corresponds to a permutation. This is precisely the feedback arc set problem, where the optimal solution has cost at most $\|B_{x^*}\|$. As the weights in the graph satisfy the triangle inequality, the algorithm returns a permutation y that is obtained by redirecting a set of edges B_y with $\|B_y\| \leq 2\|B_{x^*}\| \leq 2\Delta^*$.

▶ **Lemma 4.6.** *For the metric space* $(\mathcal{S}_n, \texttt{dist}_\tau^w)$*, any subset* $Q \subseteq \mathcal{S}_n$ *of size* 3*, and any local solution* y *obtained using the KWIK-SORT algorithm over* G_Q*, we have* $\texttt{dist}_\tau^w(x^*, y) \leq 3\Delta^*$.

4.4 Ulam

Analogously to the Kendall-tau case, we construct a tournament graph G_Q, however in this case we use Q of size 5. For the weighted Ulam distance, similar to [12], we solve the resulting feedback vertex set problem and obtain a directed acyclic graph (DAG). We return the topological order of the remaining DAG and append the removed feedback vertices arbitrarily to the end of the resulting permutation. Note that the weights are now assigned to the vertices. To solve this we use the 2-approximating feedback vertex set algorithm of Lokshtanov et al. [33]. This is a randomized algorithm that runs in time $\mathcal{O}\left(n^{17}\right)$.

For $x \in \mathcal{S}_n$, define B_x as the set of elements that are unaligned when comparing x with at least two permutations from $Q = \{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$,

$$B_x = \bigcup_{1 \leq i < j \leq 5} (I^x_{\pi_i} \cap I^x_{\pi_j}).$$

Again, using Proposition 4.2,

$$\|B_x\| \leq \sum_{1 \leq i < j \leq 5} \|I_i \cap I_j\| \leq \Delta(x). \tag{7}$$

If we remove the vertices in B_x from the graph G_Q we get a DAG. To see this, consider any two vertices in the remaining graph, say a, b. Since each of a and b is aligned with at least 4 permutations, both a and b are aligned in at least 3 permutations. Hence, the edge between a and b corresponds to the order of a and b in x. Since x is a permutation, the remaining graph is a DAG.

We now show that the weighted Ulam distance satisfies Property 2. Similar ideas were used in [12], however, we extend the proof for every two permutations in $\mathcal{S}_n$.

▶ **Lemma 4.7.** *For every* $x, y \in \mathcal{S}_n$*,* $\texttt{dist}_U^w(x, y) \leq \Delta(x) + \Delta(y)$.

Since the local solution is a 2-approximation we get that its cost is at most twice that of an optimal solution for Q which is at most $2\Delta^*$. Together with Lemma 4.7, we obtain Property 2 with $r = 5$ and $C = 3$, namely, $\texttt{dist}_U^w(x^*, y) \leq 3\Delta^*$.

References

1 Stein Aerts, Diether Lambrechts, Sunit Maity, Peter Van Loo, Bert Coessens, Frederik De Smet, Leon-Charles Tranchevent, Bart De Moor, Peter Marynen, Bassem Hassan, et al. Gene prioritization through genomic data fusion. *Nature biotechnology*, 24(5):537–544, 2006.

2 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008. doi:10.1145/1411509.1411513.

3 Noga Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006. doi:10.1137/050623905.

4 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583. ACM, 2014. doi:10.1145/2591796.2591805.

5 Christian Bachmaier, Franz J Brandenburg, Andreas Gleißner, and Andreas Hofmeier. On the hardness of maximum rank aggregation problems. *Journal of Discrete Algorithms*, 31:2–13, 2015. doi:10.1016/J.JDA.2014.10.002.

6 Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 119–126, 2010. doi:10.1145/1864708.1864733.

7 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *J. ACM*, 64(6):40:1–40:58, 2017. doi:10.1145/3125644.

8 Therese Biedl, Franz J Brandenburg, and Xiaotie Deng. On the complexity of crossings in permutations. *Discrete Mathematics*, 309(7):1813–1823, 2009. doi:10.1016/J.DISC.2007.12.088.

9 Kevin Buchin, Anne Driemel, and Martijn Struijs. On the hardness of computing an average curve. In *SWAT*, volume 162 of *LIPIcs*, pages 19:1–19:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SWAT.2020.19.

10 Amir Carmel, Debarati Das, and Tien-Long Nguyen. A scalable and unified framework to weighted rank aggregation, 2026. arXiv:2605.09653.

11 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Approximating the median under the ulam metric. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*, pages 761–775, 2021. doi:10.1137/1.9781611976465.48.

12 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Clustering permutations: New techniques with streaming applications. In *ITCS*, volume 251 of *LIPIcs*, pages 31:1–31:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.31.

13 Yuxin Chen and Changho Suh. Spectral mle: Top-k rank aggregation from pairwise comparisons. In *International Conference on Machine Learning*, pages 371–380, 2015. URL: http://proceedings.mlr.press/v37/chena15.html.

14 Vincent A Cicirello. Classification of permutation distance metrics for fitness landscape analysis. In *International Conference on Bio-inspired Information and Communication*, pages 81–97. Springer, 2019. doi:10.1007/978-3-030-24202-2_7.

15 Peter W Day. Rearrangement inequalities. *Canadian Journal of Mathematics*, 24(5):930–943, 1972.

16 Persi Diaconis and Ronald L Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 39(2):262–268, 1977.

17 Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference*, pages 613–622, 2001. doi:10.1145/371920.372165.

18 Nick Fischer, Elazar Goldenberg, Mursalin Habib, and Karthik C. S. Hardness of median and center in the ulam metric. In *ESA*, volume 351 of *LIPIcs*, pages 111:1–111:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.111.

19 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In Takao Asano, Shin-Ichi Nakano, Yoshio Okamoto, and Osamu Watanabe, editors, *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, volume 7074 of *Lecture Notes in Computer Science*, pages 374–383. Springer, 2011. doi:10.1007/978-3-642-25591-5_39.

20 MohammadTaghi Hajiaghayi, Saeed Seddighin, and Xiaorui Sun. Massively parallel approximation algorithms for edit distance and longest common subsequence. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1654–1672. SIAM, 2019. doi:10.1137/1.9781611975482.100.

21 Gary Hoppenworth, Jason W Bentley, Daniel Gibney, and Sharma V Thankachan. The fine-grained complexity of median and center string problems under edit distance. In *28th Annual European Symposium on Algorithms, ESA 2020*, 2020.

22 Sungjin Im and Mahshid Montazer Qaem. Fast and Parallelizable Ranking with Outliers from Pairwise Comparisons. In Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 173–188. Springer International Publishing, 2020. doi:10.1007/978-3-030-46150-8_11.

23 Piotr Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 428–434, 1999. doi:10.1145/301250.301366.

24 Piotr Indyk. *High-dimensional computational geometry*. stanford university, 2001.

25 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948. SIAM, 2010. doi:10.1137/1.9781611973075.76.

26 John G Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

27 Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

28 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 95–103, 2007. doi:10.1145/1250790.1250806.

29 Raivo Kolde, Sven Laur, Priit Adler, and Jaak Vilo. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*, 28(4):573–580, 2012. doi:10.1093/BIOINFORMATICS/BTR709.

30 Caitlin Kuhlman and Elke Rundensteiner. Rank aggregation algorithms for fair consensus. *Proceedings of the VLDB Endowment*, 13(12), 2020. URL: http://www.vldb.org/pvldb/vol13/p2706-kuhlman.pdf.

31 Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 19th International Conference on World Wide Web*, pages 571–580, 2010. doi:10.1145/1772690.1772749.

32 Haoming Li, Sujoy Sikdar, Rohit Vaish, Junming Wang, Lirong Xia, and Chaonan Ye. Minimizing time-to-rank: A learning and recommendation approach. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 1408–1414, 2019. doi:10.24963/IJCAI.2019/195.

33 Daniel Lokshtanov, Pranabendu Misra, Joydeep Mukherjee, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. 2-approximating feedback vertex set in tournaments. *ACM Trans. Algorithms*, 17(2):11:1–11:14, 2021. doi:10.1145/3446969.

34 Jan Pomikálek, Miloš Jakubíček, and Pavel Rychlý. Building a 70 billion word corpus of English from ClueWeb. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 502–506. European Language Resources Association (ELRA), 2012. URL: https://aclanthology.org/L12-1624/.

35 V Yu Popov. Multiple genome rearrangement by swaps and by element duplications. *Theoretical computer science*, 385(1-3):115–126, 2007. doi:10.1016/J.TCS.2007.05.029.

36 Marc Sevaux and Kenneth Sörensen. Permutation distance measures for memetic algorithms with population management. In *Proceedings of 6th Metaheuristics International Conference, MIC 2005*, pages 832–838. University of Vienna, 2005.

37 Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. URL: http://www.jstor.org/stable/1412159.

38 Charles Spearman. Footrule for measuring correlation. *British Journal of Psychology*, 2(1):89, 1906.

39 Ryuichi Takanobu, Tao Zhuang, Minlie Huang, Jun Feng, Haihong Tang, and Bo Zheng. Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning. In *The World Wide Web Conference*, pages 1771–1781, 2019. doi:10.1145/3308558.3313455.

40 H Peyton Young. Condorcet's theory of voting. *American Political science review*, 82(4):1231–1244, 1988.

41 H Peyton Young and Arthur Levenglick. A consistent extension of condorcet's election principle. *SIAM Journal on applied Mathematics*, 35(2):285–300, 1978.