algorithms

MDPI

Article

WIRE: A Weighted Item Removal Method for Unsupervised Rank Aggregation

Leonidas Akritidis † **and Panayiotis Bozanis** *,†

School of Science and Technology, International Hellenic University, 57001 Thessaloniki, Greece; lakritidis@ihu.gr
* Correspondence: pbozanis@ihu.gr
† These authors contributed equally to this work.

Abstract: Rank aggregation deals with the problem of fusing multiple ranked lists of elements into a single aggregate list with improved element ordering. Such cases are frequently encountered in numerous applications across a variety of areas, including bioinformatics, machine learning, statistics, information retrieval, and so on. The weighted rank aggregation methods consider a more advanced version of the problem by assuming that the input lists are not of equal importance. In this context, they first apply ad hoc techniques to assign weights to the input lists, and then, they study how to integrate these weights into the scores of the individual list elements. In this paper, we adopt the idea of exploiting the list weights not only during the computation of the element scores, but also to determine which elements will be included in the consensus aggregate list. More specifically, we introduce and analyze a novel refinement mechanism, called WIRE, that effectively removes the weakest elements from the less important input lists, thus improving the quality of the output ranking. We experimentally demonstrate the effectiveness of our method in multiple datasets by comparing it with a collection of state-of-the-art weighted and non-weighted techniques.

Keywords: WIRE; weighted rank aggregation; rank aggregation; item selection; unsupervised learning

Academic Editor: Steven Prestwich

Received: 29 April 2025
Revised: 10 June 2025
Accepted: 11 June 2025
Published: 12 June 2025

Citation: Akritidis, L.; Bozanis, P. WIRE: A Weighted Item Removal Method for Unsupervised Rank Aggregation. *Algorithms* **2025**, *18*, 362. https://doi.org/10.3390/a18060362

1. Introduction

Rank aggregation algorithms accept as input a set of ordered element lists submitted by a group of rankers. Then, they process these lists and return a consensus ranking with enhanced element ordering. Such problems are quite common in numerous research fields, such as bioinformatics [1,2], sports [3], recommendation systems [4,5], information retrieval [6–8], ballots [9,10], and so forth.

Most existing methods consider that input lists are of equal importance. Hence, they do not investigate issues where, for example, a list is submitted by an expert or a non-expert ranker, or a spammer submits a preference list in order to promote or downgrade specific elements. Although this non-weighted approach aligns with several application areas (e.g., fair elections), in other cases, such as collaborative filtering systems, it usually leads to output lists that suffer from bias, manipulation, and low-quality element ranking.

In contrast, the weighted rank aggregation methods take the aforementioned issues into account and attempt to assess the input list quality before they fuse them into a single ranking [11–13]. By applying unsupervised exploratory analysis techniques, they assign weights to the input lists with the aim of evaluating their significance. In the sequel, they exploit these weights during the computation of the scores of the respective list elements.

More specifically, the list weights are embodied in the individual element scores, thus affecting their ranking in the output list.

In the vast majority of cases, this is how the weighted methods exploit the learned list weights. However, the distance-based method of Akritidis et al. [13] introduced a more inspiring approach. First, an iterative algorithm was applied to estimate the importance of the input lists by measuring their distances from the generated output list. Then, the distance-based ranker weights were used in two ways as follows: (i) to compute the element scores in a manner that promotes those submitted by expert rankers, and (ii) to prevent low-quality elements submitted by non-expert rankers from entering the final aggregate list. In this context, the learned weights were used to determine the number of elements that each input list contributed to the final ranking.

The experiments of [13] demonstrated that weight-based list pruning can yield promising results, especially when the input lists are long. However, for input lists that consist of only a few items, the gains are considerably limited, or even reversed in some cases. These findings indicate that a more sophisticated approach is required to fully exploit the learned ranker weights.

To alleviate this issue, this paper introduces a novel, unsupervised method, called WIRE, that regulates the contribution of each input list in the formulation of the aggregate list. In contrast to the suggestions of [13], our strategy does not simply remove elements from the bottom of the input lists, but it carefully removes the weakest elements, with respect to their overall score, from the less important lists, with respect to their weights. In this way, it constructs aggregate lists of improved quality, even when the input preference lists are short. In addition, note that this refinement mechanism depends only on the input lists, the consensus ranking, and the weights of the involved rankers. Since this information is directly accessible either from the input or from the base aggregator, two major advantages of WIRE are derived. First, it fits into any weighted rank aggregation method and can be applied as a post-processing step. Second, its independence from individual element or list attributes (e.g., ranks, scores, lengths, etc.) renders it capable of handling partial rankings. These design choices increase our method's flexibility and applicability.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the relevant literature on weighted rank aggregation. Several preliminary elements are discussed in Section 3, whereas the proposed weighted item selection algorithm is presented and analyzed in Section 4. Section 5 describes the experimental evaluation and discusses the acquired results. The paper is concluded in Section 6 with the most significant findings and points for future research.

2. Related Work

The need for designing fair elections is quite old and has attracted the attention of researchers many decades ago. The Borda Count [14] and Condorcet criterion [15] are among the oldest rank aggregation methods that have been introduced for this purpose. Nowadays, rank aggregation has numerous applications in a wide variety of research areas, including information retrieval, bioinformatics, ballots, etc. [16–18].

In [19], Dwork et al. introduced four rank aggregation methods based on Markov chains to combine rankings coming from different search engines. Originally proposed to combat spam entries, the four methods are derived by considering different forms of the chain's transition matrix. Similarly, Ref. [20] adopted a Markov chain framework to compare the results of multiple microarray experiments expressed as ranked lists of genes.

The robust rank aggregation method of [21] proposed an unbiased probabilistic technique to process the results of various genomic analysis applications. More specifically, it

compared the actual ranking of an element with the ranking provided by a null model that dictated random ordering. This strategy renders it robust to outliers, noise, and errors.

On the other hand, the order-based methods examine the relevant rankings of the items in the input lists, assigning them scores based on their pairwise wins, ties, and losses. The Condorcet method was among the first approaches to adopt this logic [15]. Similarly, the well-known Kemeny–Young method is based on a matrix that also counts the pairwise preferences of the rankers [22,23]. Then, it uses this matrix to assign scores to each possible permutation of the input elements. Its high computational complexity—the problem is NP-Hard even for four rankers—rendered it inappropriate for datasets having many input lists. More recently, the outranking approach of [24] introduced four threshold values to quantify the concordance or discordance between the input lists.

The aforementioned methods have been proven to be quite effective in a variety of tasks. However, they do not take into consideration the level of expertise of the rankers who submit their preference lists. Consequently, they are prone to cases where a ranker may attempt to manipulate the output ranking by submitting spam entries, or simply, to rankers with different importance degrees.

The work of Pihur et al. proposed a solution that optimized the weighted distances among the input lists [25]. Based on the traditional Footrule distance, the authors established a function that combined the list weights with the individual scores that were assigned to each element by its respective rankers. However, in many cases, these scores are unknown (that is, they are not included in the input lists); therefore, the distance computation is rendered intractable.

The weighted method of Desarkar et al. modeled the problem by constructing preference relation graphs [26]. The list weights were subsequently computed by verifying the validity of several custom majoritarian rules. Furthermore, Chatterjee et al. introduced another weighted technique for crowd opinion analysis applications by adopting the principles of agglomerative clustering [27].

More recently, an iterative distance-based method called DIBRA was published in [13]. At each iteration, DIBRA computes the distances of the input lists from an aggregate list that is derived by applying a simple baseline method like the Borda Count. The lists that are proximal to the aggregate list are assigned higher weights, compared to the most distant ones. This process is repeated until all weights converge and the aggregate list is stabilized. Interestingly, Ref. [13] also introduced the idea of exploiting the learned weights to limit the contribution of the weakest rankers in the formulation of the aggregate list. In particular, the number of elements that participate in the aggregation process is determined by the weight of the respective preference list.

This simple pruning mechanism has been proven to be beneficial in several experiments that involved long input lists. However, for shorter lists, the gains were either minimized or reversed. This study introduces an item selection method that performs significantly better in all cases.

3. Weighted Rank Aggregation

A typical rank aggregation scenario involves a group of n rankers $U = \{u_1, u_2, \ldots, u_n\}$ that submits n preference lists $l^{(u_1)}, \ldots, l^{(u_n)}$ and an aggregation algorithm $\mathcal{A}$ that combines these preference lists with the aim of finding a consensus ranking $\mathcal{L}$. In this paper, we consider the most generic version of the problem, where the preference lists (i) can be of variable length $|l^{(u)}|$ and (ii) may include only a subset of the entire universe S of the elements; such lists are called partial. In contrast, full lists contain permutations of all elements of S, so they are of equal lengths.

A typical non-weighted aggregator $\mathcal{A}$ receives the preference lists as input and assigns a score value s_i to each element i, according to its ranking in the input lists, the number of pairwise wins and losses, or other criteria. Then, it outputs the unique elements of the lists sorted in score order, thus producing a consensus ranked list $\mathcal{L}$ as follows:

$$\mathcal{L} = \mathcal{A}\left(l^{(u_1)}, \ldots, l^{(u_n)}\right). \tag{1}$$

In contrast, a weighted aggregator $\mathcal{A}_w$ employs an unsupervised mechanism $\mathcal{E}$ that evaluates the importance $w^{(u)}$ of each ranker u, by taking into account its preferences and several other properties. In the sequel, it integrates these weights in the computation of the element scores as follows:

$$w^{(u_1)}, \ldots, w^{(u_n)} = \mathcal{E}\left(l^{(u_1)}, \ldots, l^{(u_n)}\right), \tag{2}$$

$$\mathcal{L} = \mathcal{A}_w\left(w^{(u_1)}, \ldots, w^{(u_n)}, l^{(u_1)}, \ldots, l^{(u_n)}\right). \tag{3}$$

In [13], the authors also introduced a post-processing technique $\mathcal{P}$ that removes the low-ranked elements from each input list based on the weights of their respective rankers as follows:

$$\mathcal{L}' = \mathcal{P}\left(\mathcal{L}, w^{(u_1)}, \ldots, w^{(u_n)}\right). \tag{4}$$

In this study, we propose a new list pruning mechanism with the aim of improving the quality of the aggregate list $\mathcal{L}'$. The block diagram of Figure 1 illustrates the sequential flow of the aforementioned procedure.

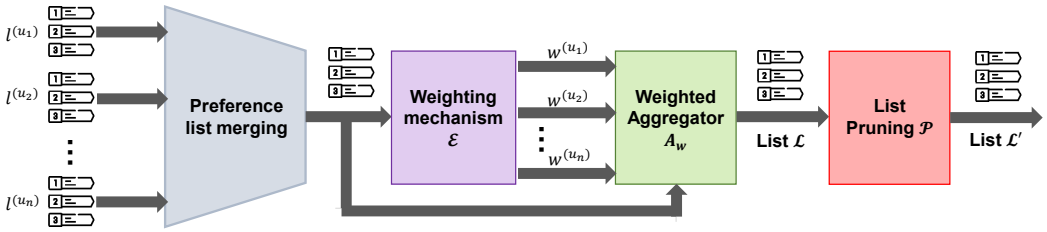

Figure 1. Schematic diagram of weighted rank aggregation with list pruning.

4. WIRE: Algorithm Design and Analysis

Algorithm description. Most weighted rank aggregation methods integrate the learned weights into the scores of the individual elements, with the sole objective of improving the quality of their produced ranking. As mentioned earlier, a representative exception is the list pruning policy of [13], which uses the learned ranker weights to remove the weakest elements from the input lists. In the sequel, we introduce a new method called *WIRE (Weighted Item Removal)* that utilizes the learned list weights to effectively determine the contribution of each input preference list in the formulation of the output list $\mathcal{L}$.

WIRE introduces a novel mechanism that identifies appropriate list elements for removal. Notably, this mechanism depends only on the input preference lists, the ranker weights, and the consensus ranking. This increases the flexibility of our algorithm, since its independence from other factors (e.g., the scores of the input list elements) enables its attachment to any weighted rank aggregation method $\mathcal{A}_w$ as a post-processing step.

On the other hand, if $\mathcal{A}_w$ cannot estimate the ranker weights in a robust manner, then the ability of WIRE to identify the weakest elements will inevitably decrease. For this reason, we present WIRE in the context of DIBRA, the distance-based weighted method that was introduced in [13]. The experiments of [13] have shown that DIBRA is superior to other weighted algorithms (e.g., [26,27]), and this was also verified in the present experiments.

Therefore, even though our method can be applied in combination with any weighted rank aggregation method, the capability of DIBRA in distinguishing the expert rankers from the non-experts renders it a suitable basis for applying WIRE.

In short, DIBRA capitalizes on the concept that the importance of a ranker is determined by the distance of its preference list from the consensus ranking $\mathcal{L}$. In this spirit, it iteratively adjusts the weight $w^{(u)}$ of a ranker u by using the following equation:

$$w_i^{(u)} = w_{i-1}^{(u)} + \exp\Big(-i \cdot d\big(l^{(u)}, \mathcal{L}_{i-1}\big)\Big), \tag{5}$$

where $w_i^{(u)}$ and $\mathcal{L}_i$ are the weight of ranker u and the aggregate list after i iterations, respectively. Moreover, d represents a function that quantifies the distance between the preference list $l^{(u)}$ and the aggregate list $\mathcal{L}$. The form of Equation (5) guarantees that the weights converge after several iterations, usually between 5 and 20.

After the voter weights have been calculated and the final aggregate list $\mathcal{L}$ has been constructed, WIRE is deployed to further improve the quality of $\mathcal{L}$. The proposed method refines the input preference lists by removing their weakest elements and operates in five phases, according to Algorithm 1.

Algorithm 1 WIRE: Weighted Item Removal

Input: Rankers $u_1 \dots u_n$, Ranker weights $w^{(u_1)}, \dots, w^{(u_n)}$, Preference lists $l^{(u_1)}, \dots, l^{(u_n)}$, number of buckets $B < n$, hyper-parameter δ_1
Output: Aggregate list $\mathcal{L}'$.

$W \leftarrow \text{sort}\big(w^{(u_1)}, \dots, w^{(u_n)}\big)$ ▷ *Sort the ranker weights*
$b, i \leftarrow 1$
while $b < B$ **do** ▷ *Compute the confidence scores of all buckets*
  $C_b \leftarrow \delta_1 + (1 - \delta_1)\exp(-(b-1)B/n)$ ▷ *Equation (6)*
  $b \leftarrow b + 1$
BucketCount $\leftarrow 1$
while $i \leq n$ **do** ▷ *Group the rankers into the buckets*
  if $i > n \cdot \text{BucketCount}/B$ **then**
    BucketCount $\leftarrow$ BucketCount $+ 1$
  $b^{(u_i)} \leftarrow$ BucketCount ▷ *Place the ranker u_i in $b^{(u_i)}$*
  $C_{b^{(u_i)}} \leftarrow C_{\text{BucketCount}}$ ▷ *u_i inherits the confidence score of its bucket $b^{(u_i)}$*
  $i \leftarrow i + 1$
for each element $\mathbf{e} \in \mathcal{L}$ **do** ▷ *Compute the preservation scores of all elements of $\mathcal{L}$*
  $P_{\mathbf{e}} \leftarrow 0, i \leftarrow 1$
  while $i < n$ **do** ▷ *Sum the confidence scores of all rankers who preferred* **e**
    if $\mathbf{e} \in l^{(u_i)}$ **then**
      $P_{\mathbf{e}} \leftarrow P_{\mathbf{e}} + C_{b^{(u_i)}}$ ▷ *Preservation score, Equation (7)*
    $i \leftarrow i + 1$
$i \leftarrow 1$
while $i \leq n$ **do** ▷ *Remove the elements with the lowest preservation scores from each list*
  $R_b^{(u_i)} \leftarrow \Big\lceil |l^{(u_i)}| C_{b^{(u_i)}} \Big\rceil$ ▷ *Protection score, Equation (8)*
  $H \leftarrow \text{BuildMinHeap}\big(l^{(u_i)}\big)$ ▷ *Build a min heap from the elements of the list*
  RemovedElements $\leftarrow 0$
  while RemovedElements $< |l^{(u_i)}| - R_b^{(u_i)}$ **do** ▷ *Remove $|l^{(u_i)}| - R_b^{(u_i)}$ elements*
    e $\leftarrow H.\text{pop}()$ ▷ *Pop the heap's head* **e**
    $l^{(u_i)} \leftarrow l^{(u_i)} -$ **e** ▷ *Remove* **e**
    RemovedElements $\leftarrow$ RemovedElements $+ 1$
  $i \leftarrow i + 1$
$\mathcal{L}' \leftarrow \mathcal{A}_w\Big(l^{(u_1)}, \dots, l^{(u_n)}\Big)$ ▷ *Rerun the weighted aggregator $\mathcal{A}_w$ on the new lists*
return $\mathcal{L}'$

Initially, the rankers are sorted in decreasing weight order. Then, they are distributed into a set of $B < n$ equally sized buckets in such a manner that each bucket contains n/B rankers. Since the ranker weights are sorted in descending order, the first bucket will contain the expert rankers, that is, those who received the n/B highest weights; the second bucket will include the rankers with lower weights, and so on. We now introduce an exponentially decaying *confidence score* for each bucket $b \in B$ as follows:

$$C_b = \delta_1 + (1 - \delta_1) \exp(-(b-1)B/n), \tag{6}$$

where $\delta_1 \in [0, 1]$ is a hyper-parameter that specifies the minimum confidence score of a bucket. The confidence scores of Equation (6) are inherited by all rankers belonging to a particular bucket, as shown in step 11 of Algorithm 1. Therefore, the rankers of the first bucket ($b = 1$) receive a confidence score $C_1 = 1$, regardless of the total number of buckets B. Similarly, all rankers in the second bucket ($b = 2$) are assigned a confidence score $S_2 = \delta_1 + (1 - \delta_1) \exp(-B/n))$, etc. Notice that $C_b \in [0, 1]$.

Subsequently, for each element $\mathbf{e} \in \mathcal{L}$ we introduce the following *preservation score*:

$$P_{\mathbf{e}} = \sum_{\forall u | \mathbf{e} \in l^{(u)}} C_{b^{(u)}} = \sum_{\forall u | \mathbf{e} \in l^{(u)}} \Big(\delta_1 + (1 - \delta_1) \exp\big(-(b^{(u)} - 1)B/n \big) \Big), \tag{7}$$

where $b^{(u)}$ denotes the bucket in which the ranker u has been placed. Equation (7) indicates that the preservation score of an item $\mathbf{e} \in \mathcal{L}$ depends on *the sum of the confidence scores of the rankers who included it in their preference lists (steps 15–17)*. Thus, if the item was preferred by numerous experts, then its preservation score will be high. In contrast, the elements submitted by a few non-experts will be assigned lower preservation scores.

Moreover, notice the independence of $P_{\mathbf{e}}$ from other parameters, e.g., the individual rankings, or the number of pairwise wins and losses, etc. This design choice renders the preservation scores robust to manipulation from spammers or low-quality preferences originating from non-expert users. The left diagram in Figure 2 illustrates an example of how confidence and preservation scores are computed for $n = 6$ rankers and $B = 3$ buckets.

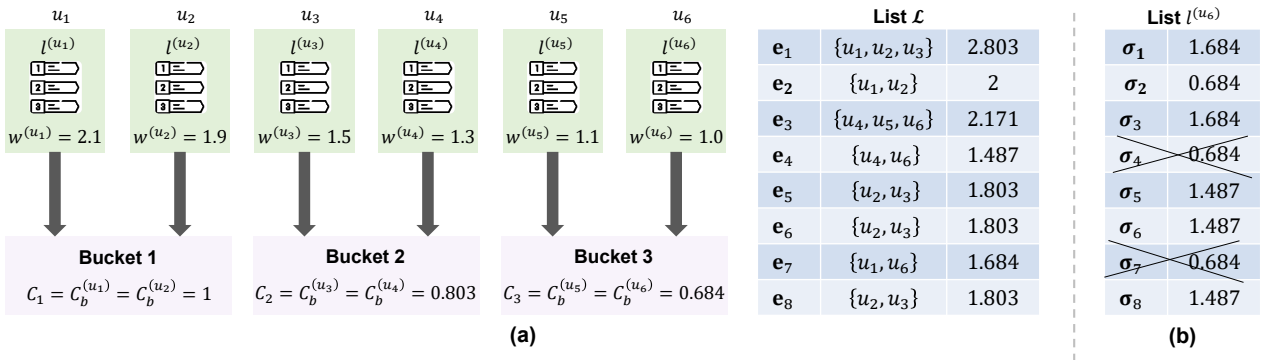

Figure 2. Indicative examples of the execution of WIRE. The left part demonstrates how the rankers are grouped into buckets and the computation of the confidence and preservation scores (**a**). The right part illustrates the removal of the weakest elements from an input preference list (**b**).

The preservation score is a measure of the importance of an element $\mathbf{e}$ and determines whether it will be included in the aggregate list or whether it will be discarded. Formally, the higher the preservation score of $\mathbf{e}$, the higher the probability that $\mathbf{e}$ will be preserved in $\mathcal{L}'$. For this reason, the *protection score* is introduced to determine the number of elements

that will be taken into account during the aggregation. Similarly to the confidence score, the protection score $R_b^{(u)}$ is defined for each ranker u according to the following equation:

$$R_b^{(u)} = \left\lceil |l^{(u)}| C_{b^{(u)}} \right\rceil. \tag{8}$$

Equation (8) defines the protection score as an integer threshold that denotes the number of elements of $l^{(u)}$ that will contribute during the aggregation process. In other words, it tells us that the $|l^{(u)}| - R_b^{(u)}$ weakest elements must be removed from $l^{(u)}$. The item removal process is shown in steps 20–28 of Algorithm 1. For each input list, we build a min-heap structure H to efficiently identify the items to be removed without having to sort $l^{(u)}$. H is built by inserting all elements of $l^{(u)}$ and keeps on its head the element with the minimum preservation score. Then, the $|l^{(u)}| - R_b^{(u)}$ items with the lowest preservation scores are indicated by performing an equal number of pop operations on H (steps 24–28). The diagram on the right in Figure 2 shows the removal of the two weakest elements from an exemplary input list based on the preservation scores of its elements.

After the removal of the weakest elements from each input list, the aggregator $\mathcal{A}_w$ is executed once more on the new lists to obtain the enhanced aggregate list $\mathcal{L}'$ (step 29).

Illustrative Examples. Figure 2 illustrates two examples of how the stages of Algorithm 1 operate in practice. The left part (a) shows the distribution of $n = 6$ rankers in $B = 3$ equally sized buckets. Initially, each ranker u submits a preference list $l^{(u)}$ and a weighted aggregator $\mathcal{A}_w$ constructs a consensus ranking $\mathcal{L}$ by simultaneously assigning weights $w^{(u_1)}, \ldots, w^{(u_6)}$ to all rankers. Setting $\delta_1 = 0.5$, the confidence scores for each bucket $b = 1, 2, 3$ are computed using Equation (6). Subsequently, the preservation scores of the elements of the aggregate list are computed with Equation (7). For example, the element $\mathbf{e}_1 \in \mathcal{L}$ appeared in the input lists of u_1, u_2, and u_3, which have been previously grouped into buckets 1, 1, and 2, respectively. Therefore, its preservation score will be computed by summing the confidence scores of the following corresponding buckets: $P_{\mathbf{e}_1} = C_{b^{(u_1)}} + C_{b^{(u_2)}} + C_{b^{(u_3)}} = 1 + 1 + 0.803 = 2.803$.

Having the preservation scores computed, we can now go back to the input lists and remove the weakest elements. First, we need to compute the protection scores for each ranker. The right part of Figure 2 displays an exemplary preference list $l^{(u_6)}$. Since u_6 has been grouped in the bucket $b = 3$, it can be derived that $C_b^{(u_6)} = 0.684$. Therefore, Equation (8) indicates that $R_b^{(u_6)} = \left\lceil |l^{(u_6)}| C_{b^{(u_6)}} \right\rceil = \lceil 8 \cdot 0.684 \rceil = 6$ elements of $l^{(u_6)}$ should be preserved, and 2 elements should be removed. According to the figure, the three elements with the lowest preservation scores are σ_2, σ_4, and σ_7. From these elements, σ_2 is preserved because it was ranked higher than the other two; σ_4 and σ_7 are eventually removed from the list.

Complexity Analysis. In the sequel, we will prove the following lemma:

Lemma 1. *The time complexity of WIRE is upper-bounded by* $O(n|\mathcal{L}| \log |\mathcal{L}|)$, *given that* $n = O(2^{|\mathcal{L}|})$, *while it needs* $\Theta(n + |\mathcal{L}|)$ *space.*

Proof. The cost of sorting the ranker weights (step 1) is $O(n \log n)$. The confidence score calculation (steps 3, 5) takes $O(B)$ time. The cost of grouping n rankers (steps 6–11) into buckets is $O(n)$. Calculating the preservation scores of all elements $\mathbf{e}$ of $\mathcal{L}$ has worst-case complexity $O(\sum_{\mathbf{e}} |u; \mathbf{e} \in l^{(u)}| = O(n|\mathcal{L}|)$, since each element may appear in all lists.

The last block of Algorithm 1 between steps 20 and 28 includes the construction of n min-heaps H ($O(n \max_u |l^{(u)}|)$ worst-case cost), and $n \max_u (|l^{(u)}| - R_b^u)$ element removals with a worst-case cost of $O(n(|\mathcal{L}| - |\mathcal{L}'|))$. Hence, the total worst-case cost of the entire block is $O(n(|\mathcal{L}| - |\mathcal{L}'|) \log \max_u |l^{(u)}|)$.

Consequently, the overall cost of WIRE is expressed as follows:

$$O(n \log n + B + n + n|\mathcal{L}| + n \max_u |l^{(u)}| + n(|\mathcal{L}| - |\mathcal{L}'|) \log \max_u |l^{(u)}|), \quad (9)$$

that can be upper-bounded by $O(n|\mathcal{L}| \log |\mathcal{L}|)$, since $B = o(n)$, $\max_u |l^{(u)}| = O(|\mathcal{L}|)$, and $n = O(2^{|\mathcal{L}|})$. Please note that this a quite pessimistic upper bound, since it assumes that every $l^{(u)}$ will be excluded from $\mathcal{L}'$ in its entirety.

The space complexity is linear to the number of rankers and the size of the initial aggregate list $\mathcal{L}$, since we need to keep all $C_{b^{(u_i)}}$ values, all P_e scores, while the maximum size of the auxiliary heap is linear to the length of the longest $l^{(u)}$, which is $O(|\mathcal{L}|)$. □

5. Experiments

This section presents the experimental evaluation of the proposed method. Initially, we describe the datasets, the comparison framework, and the measures that were used during this evaluation. In the sequel, we present and discuss the performance of WIRE against a variety of well-established rank aggregation methods in terms of both effectiveness and running times. We also present a study of how the two hyper-parameters of WIRE, B and δ_1, affect its performance. Finally, the statistical significance of the presented measurements is verified in the last part of the section.

The implementation code of the proposed method has been embodied in the FLAGR 1.0.20 and PyFLAGR 1.0.20 (https://flagr.site (accessed on 10 June 2025) libraries [28]. Both libraries are available for download from Github (https://github.com/lakritidis/FLAGR), whereas PyFLAGR can be installed from the Python Package Index (https://pypi.org/project/pyflagr/).

5.1. Datasets

The list pruning strategy of [13] was shown to be effective in scenarios that included long preference lists. However, the experiments have shown that these benefits were diminished, and in some cases, reversed in applications where the preference lists were short. For this reason, in this study, we aimed to examine the performance of WIRE by using not only multiple datasets but also, with diverse characteristics.

In this context, we synthesized six case studies with different numbers of rankers and variable list lengths. We used RASDaGen, an open source dataset generation tool (https://github.com/lakritidis/RASDaGen), to create 6 synthetic datasets that would simulate multiple real-word applications. In particular, we created datasets with long lists to simulate applications related to Bioinformatics (e.g., gene rankings) or Information Retrieval (e.g., metasearch engines). In contrast, the datasets with shorter lists can be considered as representatives of collaborative filtering systems.

In general, finding high-quality datasets with objective relevance judgments is a challenging task. Text Retrieval Conference (TREC) (https://trec.nist.gov/) satisfies these quality requirements, since it annually organizes multiple diverse tracks, accepts ranked lists from the participating groups, and employs specialists to judge the relevance of the submitted items. We used the following two real-world datasets originating from TREC: the Clinical Trials Track of 2022 (CTT22) and the NeuCLIR Technical Docs Track of 2023 (NTDT23). The first one includes 50 topics and 41 rankers, whereas the second one includes 41 topics and 51 rankers. In both datasets, the maximum length of a preference list is limited to 1000 elements.

Table 1 illustrates the attributes of our benchmark datasets. The third column denotes the number of topics for which the rankers submitted their preference lists. The fourth and fifth columns indicate the number of rankers and the length of their submitted lists,

respectively. All synthetic datasets have been made publicly available on Kaggle (https://www.kaggle.com/datasets/lakritidis/rankaggregation).

Table 1. Attributes of the benchmark datasets.

Dataset	Description	Topics $\|T\|$	n	k
MOLO	Synthetic, modest number of long lists	20	50	100
MASO	Synthetic, many short lists	20	100	30
FESO	Synthetic, few short lists	20	10	10
MOSO	Synthetic, modest number of short lists	20	50	30
MAVSO	Synthetic, many very short lists	20	500	5
FEVLO	Synthetic, few very long lists	20	10	200
CTT22	Real, 2022 TREC Clinical Trials Track	50	41	1000
NTDT23	Real, 2023 TREC NeuCLIR Technical Docs Track	41	51	1000

5.2. Comparison Framework and Evaluation Measures

The effectiveness of WIRE was compared against a wide variety of non-weighted and weighted rank aggregation methods. The first set included Borda Count and CombMNZ as they were formalized in [29], the first and the fourth Markov chain-based methods (MC1, MC4) of [19], the Markov chain framework (MCT) of [20], the Robust Rank Aggregation (RRA) method of [21], the Outranking Approach (OA) of [24], the traditional Condorcet [15], and Copeland Winners [30].

The second set included the following four state-of-the-art weighted methods: DIBRA with and without list pruning (termed DIBRA-P and DIBRA, respectively) [13], the weighted approach of [26] based on Preference Relation Graphs (PRGs), and the Agglomerative Aggregation Method (AAM) of [27]. In all methods, we used the same hyper-parameter values as those suggested in the respective studies. Our item removal method was tested in combination with DIBRA, and we refer to it as DIBRA-WIRE in the discussion that follows. In all experiments, we kept the values of the two hyper-parameters of WIRE constant. Therefore, the number of buckets was fixed to $B = 5$, and we set $\delta_1 = 0.5$.

The quality of the output lists was measured by employing the following three widespread IR measures: Precision, Normalized Discounted Cumulative Gain (nDCG), and Mean Average Precision (MAP). The first two are computed at specific cut-off points of the aggregate list $\mathcal{L}$, and we refer to them as P@k and N@k, respectively. Precision@k is defined as follows:

$$\text{P@}k = \frac{1}{k} \sum_{i=1}^{k} \text{rel}(\mathcal{L}_i), \tag{10}$$

where $\text{rel}(\mathcal{L}_i)$ is a binary indicator of the relevance of the i-th element of $\mathcal{L}$ (1/0 for relevant/non-relevant).

On the other hand, nDCG is defined as the Discounted Cumulative Gain (DCG) of $\mathcal{L}$ divided by the Discounted Cumulative Gain of an imaginary ideal list that has all relevant elements ranked at its highest positions as follows:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{\text{rel}(\mathcal{L}_i)} - 1}{\log_2(k+1)}, \tag{11}$$

$$\text{N@}k = \frac{\text{DCG@}k}{\text{iDCG@}k}. \tag{12}$$

Finally, Mean Average Precision (MAP) is defined as the mean of the average precision scores among a set of T topics as follows:

$$\text{MAP} = \frac{1}{|T|}\sum_{t=1}^{|T|}\overline{P}(t) = \frac{1}{|T|}\sum_{t=1}^{|T|}\frac{1}{R_t}\sum_{k=1}^{|\mathcal{L}|}\text{rel}(\mathcal{L}_k)\text{P@}k, \tag{13}$$

where $\mathcal{L}_i$ is the i-th element of the aggregate list $\mathcal{L}$, P@k denotes the Precision measured at $\mathcal{L}_k$, and R_t is the total number of relevant entries for the topic $t \in T$.

5.3. Results and Discussion

In this subsection, we present the results of the experimental evaluation of WIRE. In Tables 2–4, we report the values of the aforementioned evaluation measures for the 8 datasets of Table 1. More specifically, the second column illustrates the MAP values achieved by the examined methods, the next 5 columns hold the Precision values for the top-5 elements of the produced aggregate list $\mathcal{L}$, whereas the last 5 columns denote the nDCG values also for the top-5 elements of $\mathcal{L}$.

Table 2. Performance evaluation of various rank aggregation methods in the MOLO, MASO, and FESO datasets.

MOLO	**MAP**	**Precision**					**NDCG**				
		P@1	**P@2**	**P@3**	**P@4**	**P@5**	**N@1**	**N@2**	**N@3**	**N@4**	**N@5**
Borda Count	0.240	0.250	0.150	0.167	0.175	0.170	0.250	0.173	0.179	0.183	0.178
CombMNZ	0.238	0.200	0.225	0.200	0.188	0.170	0.200	0.219	0.203	0.194	0.182
Condorcet	0.239	0.300	0.175	0.167	0.188	0.170	0.300	0.203	0.191	0.201	0.188
Copeland	0.238	0.200	0.175	0.150	0.188	0.160	0.200	0.181	0.162	0.185	0.167
Outranking	0.238	0.300	0.200	0.167	0.163	0.190	0.300	0.223	0.194	0.186	0.201
MC1	0.249	0.250	0.250	0.217	0.263	0.250	0.250	0.250	0.227	0.256	0.248
MC4	0.249	0.250	0.250	0.217	0.263	0.250	0.250	0.250	0.227	0.256	0.248
MCT	0.238	0.000	0.100	0.167	0.212	0.210	0.000	0.077	0.130	0.167	0.171
RRA	0.240	0.100	0.100	0.167	0.188	0.190	0.100	0.100	0.147	0.164	0.169
PRG	0.239	0.250	0.200	0.167	0.175	0.170	0.250	0.211	0.185	0.188	0.183
AAM	0.136	0.300	0.275	0.233	0.275	0.260	0.300	0.281	0.250	0.275	0.265
DIBRA	0.244	0.300	0.225	0.200	0.237	0.220	0.300	0.242	0.220	0.242	0.230
DIBRA-P	0.245	0.200	0.225	0.217	0.237	0.220	0.200	0.219	0.215	0.229	0.219
DIBRA-WIRE	0.250	0.350	0.325	0.233	0.263	0.230	0.350	0.331	0.265	0.279	0.256
MASO	**MAP**	**Precision**					**NDCG**				
		P@1	**P@2**	**P@3**	**P@4**	**P@5**	**N@1**	**N@2**	**N@3**	**N@4**	**N@5**
Borda Count	0.314	0.300	0.275	0.317	0.287	0.290	0.300	0.281	0.309	0.290	0.292
CombMNZ	0.317	0.350	0.275	0.300	0.312	0.320	0.350	0.292	0.306	0.313	0.318
Condorcet	0.311	0.350	0.275	0.333	0.300	0.300	0.350	0.292	0.329	0.307	0.306
Copeland	0.312	0.300	0.250	0.300	0.287	0.310	0.300	0.261	0.294	0.286	0.301
Outranking	0.317	0.300	0.250	0.317	0.325	0.300	0.300	0.261	0.306	0.313	0.298
MC1	0.319	0.350	0.325	0.317	0.325	0.330	0.350	0.331	0.323	0.328	0.331
MC4	0.319	0.350	0.325	0.317	0.325	0.330	0.350	0.331	0.323	0.328	0.331
MCT	0.320	0.150	0.275	0.317	0.300	0.290	0.150	0.247	0.283	0.277	0.274
RRA	0.318	0.300	0.325	0.300	0.287	0.310	0.300	0.319	0.303	0.294	0.308
PRG	0.316	0.350	0.225	0.317	0.338	0.330	0.350	0.253	0.311	0.326	0.323
AAM	0.182	0.150	0.300	0.350	0.338	0.300	0.150	0.266	0.309	0.308	0.287
DIBRA	0.324	0.400	0.300	0.317	0.312	0.300	0.400	0.323	0.329	0.324	0.314
DIBRA-P	0.323	0.350	0.325	0.333	0.287	0.270	0.350	0.331	0.335	0.304	0.290
DIBRA-WIRE	0.327	0.400	0.250	0.333	0.300	0.320	0.400	0.284	0.335	0.312	0.324

Table 2. *Cont.*

FESO	MAP	Precision					NDCG				
		P@1	P@2	P@3	P@4	P@5	N@1	N@2	N@3	N@4	N@5
Borda Count	0.288	0.150	0.100	0.117	0.125	0.130	0.150	0.150	0.190	0.222	0.255
CombMNZ	0.298	0.200	0.125	0.117	0.125	0.130	0.200	0.181	0.202	0.232	0.264
Condorcet	0.228	0.100	0.125	0.117	0.138	0.140	0.100	0.151	0.175	0.228	0.268
Copeland	0.263	0.100	0.075	0.133	0.138	0.130	0.100	0.100	0.177	0.207	0.239
Outranking	0.289	0.200	0.125	0.117	0.113	0.120	0.200	0.181	0.202	0.210	0.243
MC1	0.377	0.250	0.200	0.167	0.163	0.160	0.250	0.282	0.293	0.326	0.344
MC4	0.268	0.100	0.100	0.117	0.138	0.130	0.100	0.132	0.159	0.221	0.243
MCT	0.291	0.100	0.100	0.100	0.125	0.150	0.100	0.151	0.170	0.212	0.278
RRA	0.296	0.150	0.125	0.133	0.138	0.140	0.150	0.162	0.216	0.248	0.266
PRG	0.291	0.150	0.125	0.117	0.125	0.130	0.150	0.169	0.193	0.225	0.257
AAM	0.259	0.250	0.175	0.150	0.138	0.120	0.250	0.262	0.278	0.322	0.343
DIBRA	0.332	0.150	0.175	0.150	0.150	0.140	0.150	0.232	0.247	0.270	0.304
DIBRA-P	0.272	0.100	0.200	0.150	0.150	0.140	0.100	0.226	0.219	0.258	0.272
DIBRA-WIRE	0.399	0.200	0.250	0.200	0.175	0.170	0.200	0.333	0.343	0.361	0.334

Table 3. Performance evaluation of various rank aggregation methods in the MOSO, MAVSO, and FEVLO datasets.

MOSO	MAP	Precision					NDCG				
		P@1	P@2	P@3	P@4	P@5	N@1	N@2	N@3	N@4	N@5
Borda Count	0.507	0.350	0.450	0.400	0.425	0.440	0.350	0.427	0.397	0.415	0.426
CombMNZ	0.508	0.350	0.450	0.400	0.425	0.440	0.350	0.427	0.397	0.415	0.426
Condorcet	0.512	0.450	0.500	0.483	0.463	0.490	0.450	0.489	0.480	0.466	0.484
Copeland	0.516	0.450	0.475	0.500	0.500	0.520	0.450	0.469	0.488	0.490	0.505
Outranking	0.507	0.350	0.450	0.400	0.425	0.440	0.350	0.427	0.397	0.415	0.426
MC1	0.512	0.400	0.475	0.467	0.500	0.500	0.400	0.458	0.456	0.480	0.483
MC4	0.516	0.450	0.450	0.500	0.487	0.470	0.450	0.450	0.485	0.479	0.469
MCT	0.494	0.450	0.425	0.417	0.450	0.440	0.450	0.431	0.423	0.445	0.439
RRA	0.494	0.300	0.350	0.400	0.438	0.450	0.300	0.339	0.377	0.406	0.418
PRG	0.507	0.350	0.450	0.400	0.425	0.440	0.350	0.427	0.397	0.415	0.426
AAM	0.306	0.600	0.500	0.467	0.463	0.450	0.600	0.523	0.494	0.486	0.475
DIBRA	0.517	0.400	0.400	0.450	0.500	0.490	0.400	0.400	0.435	0.471	0.469
DIBRA-P	0.499	0.450	0.400	0.417	0.425	0.470	0.450	0.411	0.420	0.425	0.455
DIBRA-WIRE	0.521	0.450	0.425	0.450	0.487	0.490	0.450	0.431	0.447	0.473	0.476
MAVSO	**MAP**	**Precision**					**NDCG**				
		P@1	P@2	P@3	P@4	P@5	N@1	N@2	N@3	N@4	N@5
Borda Count	0.400	0.400	0.275	0.233	0.250	0.230	0.400	0.323	0.294	0.322	0.328
CombMNZ	0.403	0.400	0.275	0.250	0.225	0.220	0.400	0.323	0.306	0.306	0.320
Condorcet	0.365	0.350	0.275	0.250	0.250	0.230	0.350	0.304	0.287	0.306	0.311
Copeland	0.362	0.350	0.275	0.250	0.237	0.230	0.350	0.304	0.287	0.296	0.310
Outranking	0.416	0.400	0.350	0.283	0.275	0.260	0.400	0.381	0.338	0.352	0.362
MC1	0.379	0.300	0.350	0.283	0.263	0.240	0.300	0.351	0.318	0.325	0.325
MC4	0.379	0.300	0.350	0.283	0.263	0.240	0.300	0.351	0.318	0.325	0.325
MCT	0.281	0.150	0.200	0.200	0.200	0.190	0.150	0.189	0.191	0.195	0.196
RRA	0.400	0.400	0.275	0.250	0.225	0.210	0.400	0.323	0.306	0.306	0.311
PRG	0.403	0.400	0.275	0.250	0.225	0.230	0.400	0.323	0.306	0.306	0.328
AAM	0.188	0.250	0.250	0.250	0.200	0.210	0.250	0.262	0.267	0.240	0.258
DIBRA	0.405	0.350	0.275	0.300	0.263	0.270	0.350	0.292	0.332	0.325	0.351
DIBRA-P	0.296	0.200	0.200	0.183	0.175	0.210	0.200	0.200	0.188	0.182	0.219
DIBRA-WIRE	0.414	0.350	0.350	0.317	0.287	0.280	0.350	0.362	0.357	0.355	0.371

Table 3. *Cont.*

FEVLO	MAP	Precision					NDCG				
		P@1	P@2	P@3	P@4	P@5	N@1	N@2	N@3	N@4	N@5
Borda Count	0.785	0.700	0.800	0.767	0.787	0.780	0.700	0.777	0.759	0.774	0.771
CombMNZ	0.785	0.700	0.775	0.767	0.787	0.780	0.700	0.758	0.756	0.772	0.769
Condorcet	0.785	0.650	0.750	0.800	0.800	0.800	0.650	0.727	0.768	0.773	0.777
Copeland	0.786	0.700	0.800	0.800	0.800	0.800	0.700	0.777	0.783	0.786	0.787
Outranking	0.785	0.700	0.800	0.767	0.787	0.780	0.700	0.777	0.759	0.774	0.771
MC1	0.785	0.800	0.850	0.817	0.787	0.780	0.800	0.839	0.818	0.798	0.792
MC4	0.786	0.700	0.750	0.800	0.800	0.790	0.700	0.739	0.777	0.780	0.776
MCT	0.785	0.800	0.825	0.750	0.738	0.760	0.800	0.842	0.785	0.771	0.781
RRA	0.783	0.800	0.825	0.800	0.738	0.760	0.800	0.819	0.803	0.761	0.772
PRG	0.785	0.700	0.800	0.767	0.787	0.780	0.700	0.777	0.759	0.774	0.771
AAM	0.380	0.750	0.700	0.750	0.762	0.780	0.750	0.711	0.744	0.753	0.766
DIBRA	0.785	0.750	0.775	0.783	0.787	0.790	0.750	0.769	0.777	0.780	0.783
DIBRA-P	0.783	0.800	0.800	0.867	0.850	0.850	0.800	0.900	0.877	0.864	0.862
DIBRA-WIRE	0.789	0.800	0.850	0.800	0.762	0.730	0.800	0.761	0.794	0.770	0.747

Table 4. Performance evaluation of various rank aggregation methods in the CTT22 and NTDT23 datasets.

CTT22	MAP	Precision					NDCG				
		P@1	P@2	P@3	P@4	P@5	N@1	N@2	N@3	N@4	N@5
Borda Count	0.380	0.760	0.710	0.707	0.690	0.684	0.560	0.557	0.566	0.558	0.556
CombMNZ	0.378	0.760	0.710	0.700	0.685	0.680	0.560	0.557	0.561	0.554	0.552
Condorcet	0.418	0.700	0.690	0.733	0.730	0.720	0.567	0.559	0.561	0.563	0.552
Copeland	0.418	0.720	0.780	0.780	0.795	0.764	0.567	0.559	0.561	0.563	0.552
Outranking	0.401	0.720	0.710	0.720	0.725	0.716	0.587	0.589	0.578	0.578	0.572
MC1	0.267	0.700	0.710	0.653	0.645	0.640	0.567	0.569	0.537	0.527	0.518
MC4	0.133	0.240	0.220	0.207	0.195	0.180	0.227	0.206	0.184	0.173	0.159
MCT	0.294	0.720	0.710	0.653	0.685	0.680	0.315	0.569	0.564	0.527	0.552
RRA	0.411	0.740	0.760	0.740	0.730	0.728	0.647	0.652	0.632	0.612	0.605
PRG	–	–	–	–	–	–	–	–	–	–	–
AAM	–	–	–	–	–	–	–	–	–	–	–
DIBRA	0.422	0.700	0.690	0.733	0.725	0.708	0.567	0.559	0.564	0.564	0.546
DIBRA-P	0.422	0.720	0.700	0.733	0.725	0.708	0.560	0.565	0.564	0.560	0.542
DIBRA-WIRE	0.424	0.760	0.800	0.780	0.785	0.768	0.653	0.664	0.644	0.634	0.617
NTDT23	**MAP**	**Precision**					**NDCG**				
		P@1	P@2	P@3	P@4	P@5	N@1	N@2	N@3	N@4	N@5
Borda Count	0.306	0.512	0.463	0.422	0.402	0.400	0.366	0.376	0.360	0.355	0.361
CombMNZ	0.300	0.512	0.463	0.423	0.390	0.390	0.366	0.376	0.360	0.350	0.357
Condorcet	0.355	0.659	0.537	0.520	0.470	0.434	0.596	0.491	0.467	0.455	0.436
Copeland	0.355	0.659	0.537	0.512	0.470	0.434	0.596	0.491	0.467	0.455	0.436
Outranking	0.348	0.585	0.524	0.480	0.476	0.444	0.460	0.453	0.425	0.426	0.415
MC1	0.222	0.439	0.378	0.358	0.323	0.307	0.355	0.339	0.318	0.300	0.296
MC4	0.304	0.610	0.561	0.504	0.445	0.410	0.547	0.513	0.467	0.436	0.419
MCT	0.253	0.439	0.444	0.388	0.373	0.307	0.375	0.359	0.324	0.311	0.296
RRA	0.355	0.634	0.585	0.504	0.470	0.434	0.592	0.529	0.466	0.447	0.432
PRG	0.308	0.512	0.488	0.439	0.402	0.385	0.366	0.395	0.374	0.363	0.358
AAM	–	–	–	–	–	–	–	–	–	–	–
DIBRA	0.356	0.659	0.537	0.504	0.457	0.434	0.575	0.472	0.451	0.439	0.431
DIBRA-P	0.355	0.659	0.549	0.504	0.457	0.434	0.575	0.482	0.453	0.441	0.434
DIBRA-WIRE	0.360	0.659	0.549	0.504	0.470	0.434	0.596	0.491	0.454	0.442	0.427

Now, let us discuss these numbers. At first, the results indicate that DIBRA was the most effective weighted rank aggregation method, achieving the highest MAP scores among the other two methods, PRG and AAM. Interestingly, in all eight cases, our proposed WIRE method managed to further improve the performance of DIBRA by a margin between 1% and 20% (in the FESO dataset). In fact, in most of the examined benchmark datasets, the combination of DIBRA and WIRE outperformed all the competitive aggregation methods, weighted or not.

In the MOLO dataset with the long lists of 100 items, the MAP gain of DIBRA-WIRE compared to DIBRA was roughly 2% (0.250 vs. 0.244). In contrast, the simple list pruning method of [13] offered only infinitesimal improvements in terms of MAP scores. In addition, its effect on the quality of the top-5 elements of the aggregate list was negative in terms of both Precision and nDCG. MC1 and MC4 were also quite effective in this dataset.

The next three datasets, MASO, FESO, and MOSO, were of particular interest, because they involved short lists, and the simple pruning method of [13] is known to perform poorly in such cases. In this experiment, we examined three different scenarios, where variable populations of rankers (i.e., 100, 10, and 50, respectively) submitted short preference lists comprising 30, 10, and 30 items, respectively.

The results were particularly satisfactory in all these scenarios. Compared to DIBRA, our proposed item selection method achieved superior performance in terms of MAP, Precision, and nDCG. For example, the MAP improvements over DIBRA were roughly 1% for MASO and MOSO, and 20% for FESO. Improvements of similar magnitudes were also observed for Precision and nDCG, especially for P@1 and N@1. As it was expected, the Mean Average Precision of DIBRA-P was slightly worse compared to that of DIBRA for MASO (0.323 vs. 0.324) and significantly worse in FESO (0.272 vs. 0.332). DIBRA and DIBRA-WIRE were also superior to all the other aggregation methods in all three datasets. The only exception to this observation was MC1 in the FESO dataset, which was superior to DIBRA but inferior to the combination of DIBRA-WIRE.

The fifth case study, abbreviated MAVSO, introduced a scenario that resembles that of a recommender system, namely, numerous rankers submitting very short preference lists of 5 elements. The Outranking Approach of [24] exhibited the best performance in this test, achieving the highest MAP (0.416), P@1 = 0.400, and N@1 = 0.400. Our proposed DIBRA-WIRE method was the second best method, achieving a slightly worse MAP (0.414), P@1 = 0.350, and N@1 = 0.350. The original DIBRA without list pruning outperformed the other two weighted methods, PRG and AAM, but the application of the simple list pruning method of [13] had a strong negative impact on its performance.

The sixth test resembled that of a metasearch engine, with few rankers submitting very long ranked lists of 200 items. Once again, DIBRA-WIRE achieved top performance in terms of MAP (0.789), P@1 = 0.800, and N@1 = 0.800. The method of Copeland Winners and MC4 scored the second highest MAP (i.e., 0.786), but in terms of Precision and nDCG, the aggregate lists created by MCT, RRA, and DIBRA-P were of higher quality. Regarding the weighted methods, DIBRA was more effective than PRG and AAM in terms of Precision and nDCG.

In general, in the six synthetic datasets, the Agglomerative AAM method achieved very low MAP values, but it managed to output decent top-5 rankings. This behavior indicates its weakness in generating high-quality rankings in their entirety. The Preference Relations Graph method was superior to AAM but inferior to DIBRA. On the other hand, the quite old Markov chain-based algorithms were quite strong opponents on datasets with long preference lists (namely, MOLO and FEVLO), but their performance degraded when they were applied to short lists. On average, the Outranking Approach achieved higher MAP values than the Markov Chain methods, but their top-5 rankings were of inferior

quality. As mentioned earlier, DIBRA was the most effective rank aggregation method, and our WIRE post-processing step further improved its performance.

Regarding the two real-world datasets, namely, CTT22 and NTDT23, the proposed method was again proved to be beneficial, since it managed to improve the retrieval effectiveness of the baseline DIBRA, even by small margins. On the other hand, the list pruning algorithm of [13] had almost no visible impact in both cases. The order-based methods (i.e., Condorcet, Copeland, and the Outranking Approach) were quite strong opponents to DIBRA and DIBRA-WIRE, producing very qualitative top-5 aggregate lists. In contrast, the three Markov chain-based methods (that performed quite well in the six synthetic datasets) were both ineffective and slow in these two experiments.

The other two weighted aggregation methods had significant difficulties in completing these tests. More specifically, AAM failed to generate consensus rankings in reasonable time, since it required more than 1 h to process one topic from each dataset. In general, AAM was by far the slowest method among all, due to its computationally expensive hierarchical nature. For this reason, and since the TREC datasets were significantly larger than the synthetic ones, Table 4 does not contain results from AAM. On the other hand, PRG is a graph-based memory-intensive algorithm and that became a major bottleneck in the two large TREC datasets. Therefore, a memory starvation error prevented its execution in CTT22. Nevertheless, it managed to complete the aggregation task in all 41 topics of NTDT23, achieving a MAP value of 0.307, 17% lower than that of DIBRA.

5.4. Execution Times

In this subsection we examine how WIRE affects the execution time of a weighted rank aggregation method and particularly, DIBRA. Table 5 displays the running times of the involved rank aggregation methods in the benchmark datasets of Table 1. The presented values reveal the running times of each method per topic in milliseconds (recall that each dataset comprises multiple topics).

Table 5. Execution times (in milliseconds per query) of various rank aggregation methods on the benchmark datasets of Table 1.

Method	MOLO	MASO	FESO	MOSO	MAVSO	FEVLO	CTT22	NTDT23
Borda Count	5.07	2.91	0.01	1.63	2.97	0.05	41.2	22.7
CombMNZ	5.04	2.93	0.01	1.62	2.85	0.05	41.0	25.4
Condorcet	18.88	7.40	0.01	1.93	4.05	0.10	10×10^3	8195
Copeland	17.64	6.89	0.01	1.88	3.80	0.10	10×10^3	8219
Outranking	31.28	10.26	0.01	2.22	4.57	0.16	26×10^3	23×10^3
MC1	28.81	10.29	0.01	2.09	4.94	0.13	23×10^3	19×10^3
MC4	29.06	10.24	0.01	2.07	4.84	0.15	105×10^3	68×10^3
MCT	29.69	10.33	0.01	2.04	5.00	0.15	165×10^3	109×10^3
RRA	6.11	3.70	0.01	1.83	4.84	0.05	65.67	32.90
PRG	53.51	17.02	0.01	2.67	6.68	0.33	–	98×10^3
AAM	2096.71	1940.83	0.11	141.86	14×10^3	0.79	>1 h	>1 h
DIBRA	5.98	3.87	0.01	2.05	4.17	0.05	82.33	53.94
DIBRA-P	6.45	4.24	0.01	2.06	4.59	0.06	87.26	46.69
DIBRA-WIRE	8.11	5.05	0.01	2.67	5.89	0.07	112.40	78.51

These measurements indicate that WIRE imposes only small (and in some cases, infinitesimal) delays in the aggregation process. More specifically, in the six synthetic datasets, Algorithm 1 appended on average only 0.5–2.2 milliseconds to the processing of each query, demonstrating its high efficiency. In fact, DIBRA-WIRE was much faster than (i) the other weighted algorithms, PRG and AAM; (ii) all the order-based techniques

(Condorcet and Copeland methods, Outranking approach); and (iii) all the Markov chain-based methods (MC1, MC4, MCT). In contrast, it was slightly slower only than simple linear combination methods like Borda Count and CombMNZ.

As mentioned earlier, in the large TREC datasets, AAM failed to construct aggregate lists in reasonable time (more than one hour per topic). Moreover, PRG consumed all the available memory of our 32 GB workstation in CTT22, also failing to complete the aggregation task. In NTDT23, it was much slower than DIBRA, putting its usefulness into question. This conclusion applies to all the order-based and Markov chain-based methods. Despite their effectiveness in some experiments, the average topic processing times can be 3–5 orders of magnitude larger than those of Borda Count, DIBRA, DIBRA-WIRE, etc.

5.5. Statistical Significance Tests

The reliability of the aforementioned measurements was checked by applying two statistical significance tests. In particular, we executed the Friedman non-parametric statistical test in the measurements of MAP and P@1 of all methods. The results of the tests were equal to 6.9×10^{-4} and 1.5×10^{-4}, respectively, indicating that the measurements' distributions exhibit statistically significant discrepancies and that we can reject the null hypothesis.

We also applied the Wilcoxon signed-rank test to examine the pairwise statistical significance of the performances of DIBRA, DIBRA-P, and DIBRA-WISE in terms of Mean Average Precision. The results are presented in Table 6 and indicate that the MAP differences between DIBRA-WIRE and DIBRA are more significant than between DIBRA-P and DIBRA.

Table 6. Statistical significance of the MAP measurements of DIBRA-WIRE against DIBRA and DIBRA-P. The presented p-values have been obtained by executing the Wilcoxon post hoc signed-rank test.

Method	**DIBRA-WIRE (MAP)**
DIBRA-WIRE vs. DIBRA	0.0078125
DIBRA-WIRE vs. DIBRA-P	0.0078125
DIBRA vs. DIBRA-P	0.0141051

5.6. Hyper-Parameter Study

The proposed item removal method introduces the following two hyper-parameters: the number of buckets B in which the rankers are grouped, and the parameter δ_1 of Equation (6). In this subsection, we study how these parameters affect the performance of DIBRA-WIRE, and why the setting of $B = 5$ and $\delta_1 = 0.5$ that was applied in the previous experiment is a choice that, in general, leads to good results.

Our methodology for this study dictated the parallel variation of the values of B and δ_1 and the independent measurement of MAP in each case. More specifically, for each one of the eight benchmark datasets, we modified B in the range $[2, 12]$ by taking integer steps of 1 at each time (namely, we tested 11 values). Simultaneously we modified d_1 in the range $[0, 1]$ by taking steps of $0, 1$ at each iteration, also considering 11 values. In other words, we conducted 121 measurements of MAP for each dataset.

The results are illustrated in the heatmaps of Figures 3 and 4. Each heatmap represents the MAP measurements for a different benchmark dataset. The horizontal and vertical axes depict the values of $B \in [2, 12]$ and $\delta_1 \in [0, 1]$, respectively. The darker background colors denote higher MAP values, with the black rectangles revealing top performance.

A careful observation of these heatmaps reveals that our choice of $B = 5$ and $\delta_1 = 0.5$ yields satisfactory results in all cases. Of course, there are combinations that "optimize" the performance, but this is not consistent. For example, setting $B = 4$ and $\delta_1 = 0.3$ yielded the best performance in terms of MAP in the MASO dataset. However, this setting does

not work well in the FESO dataset. Similarly, $B = 11$ and $\delta_1 = 0$ was the best combination for MOSO, but it was also among the worst in the MOLO, MASO, and FESO datasets. For FESO, the best setting was $B = 12$ and $\delta_1 = 0.6$; a rather disappointing choice for MOLO and MASO.

We also constructed similar heatmaps for all Precision and nDCG values at the top-5 elements of each aggregate list $\mathcal{L}$. Due to space restrictions, we cannot illustrate 80 such diagrams here. Instead, we indicatively choose to depict the fluctuation of N@5 against B and δ_1 in Figures 5 and 6. The results of this exhaustive study verify the conclusions of our previous experiment. Namely, there are multiple combinations of B and δ_1 that yield decent results, but similarly to the MAP case, there is no "golden rule" achieving top performance in all scenarios. In contrast, one that consistently performs well is $B = 5$ and $\delta_1 = 0.5$.

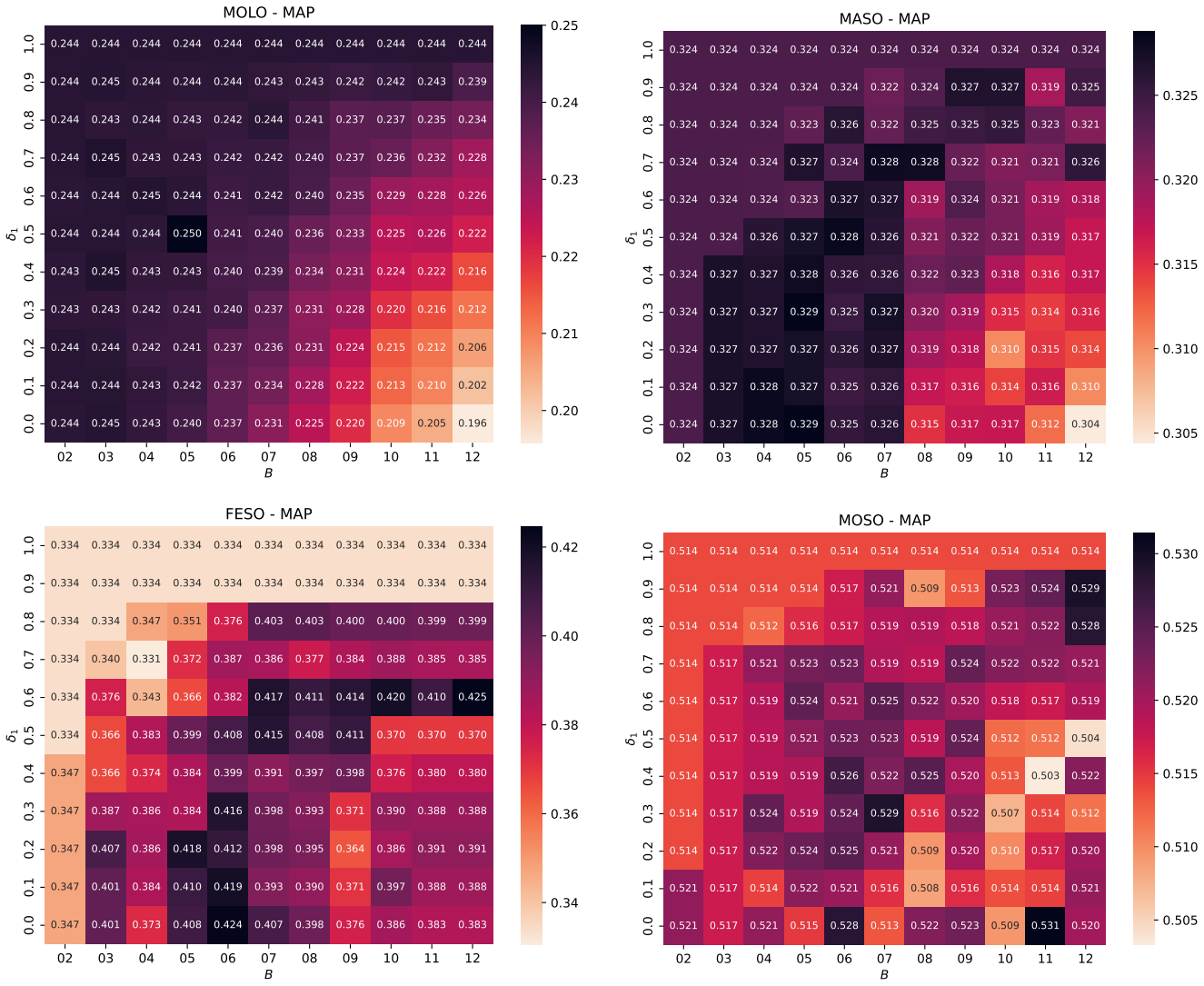

Figure 3. Hyper-parameter study of DIBRA-WIRE for the MOLO, MASO, FESO, and MOSO datasets. The heatmaps illustrate the fluctuation of Mean Average Precision for variable number of buckets B (horizontal axis) and variable δ_1 values (in the range $[0, 1]$).

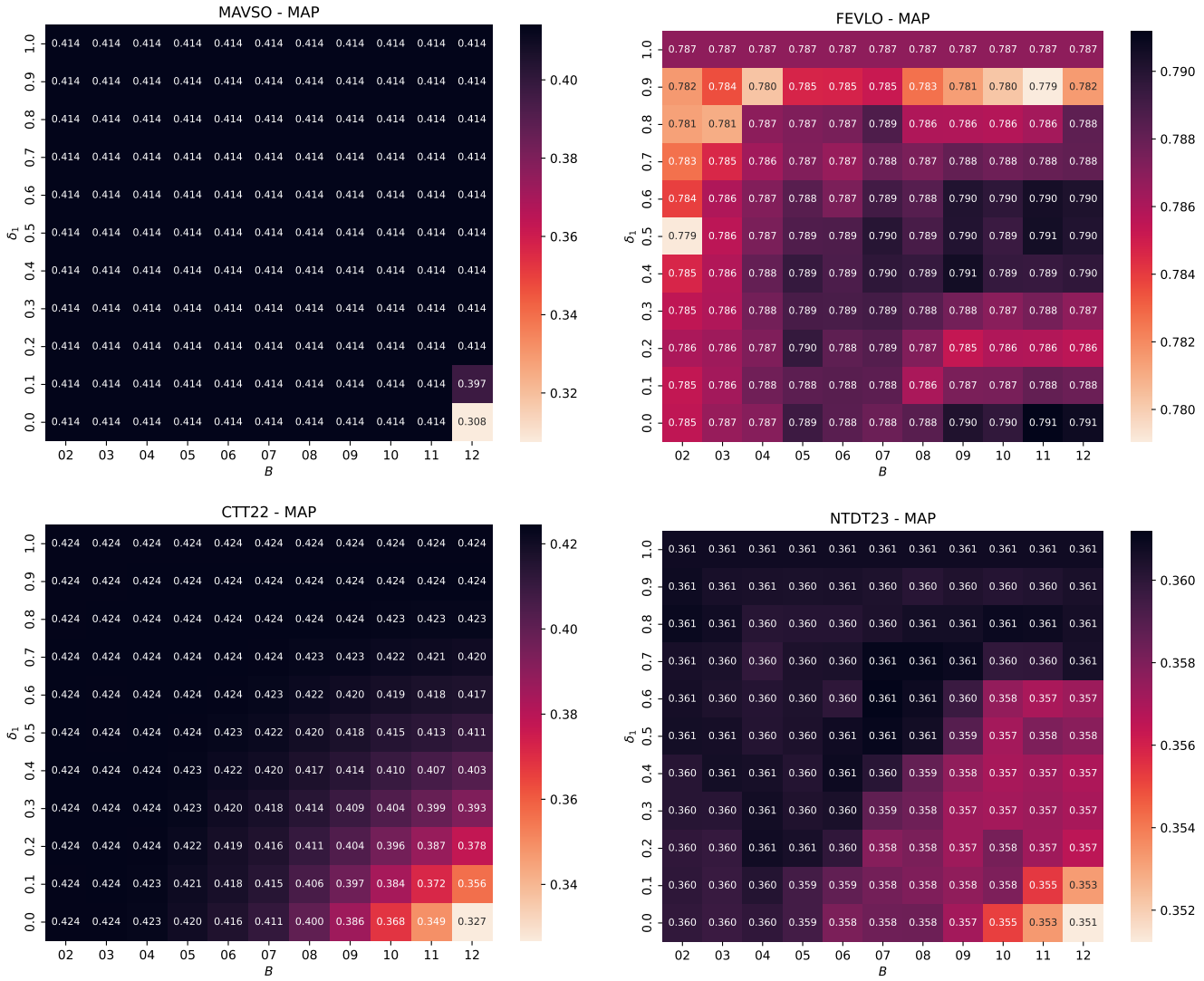

Figure 4. Hyper-parameter study of DIBRA-WIRE for the MAVSO, FEVLO, CTT22, and NTDT23 datasets. The heatmaps illustrate the fluctuation of Mean Average Precision for a variable number of buckets B (horizontal axis) and variable δ_1 values (in the range $[0, 1]$).

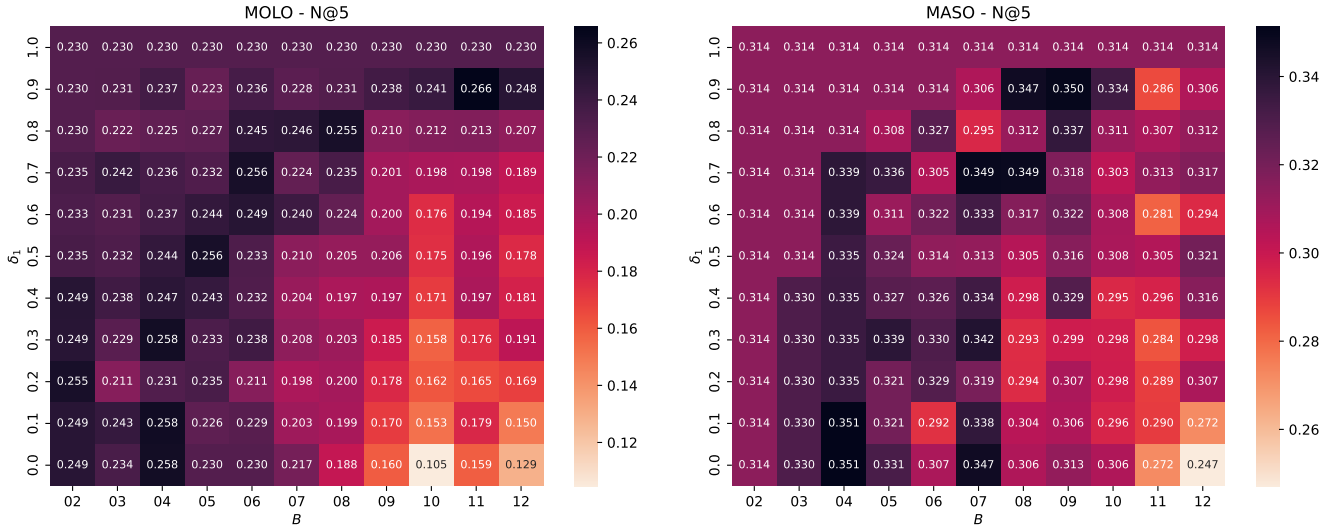

Figure 5. *Cont.*

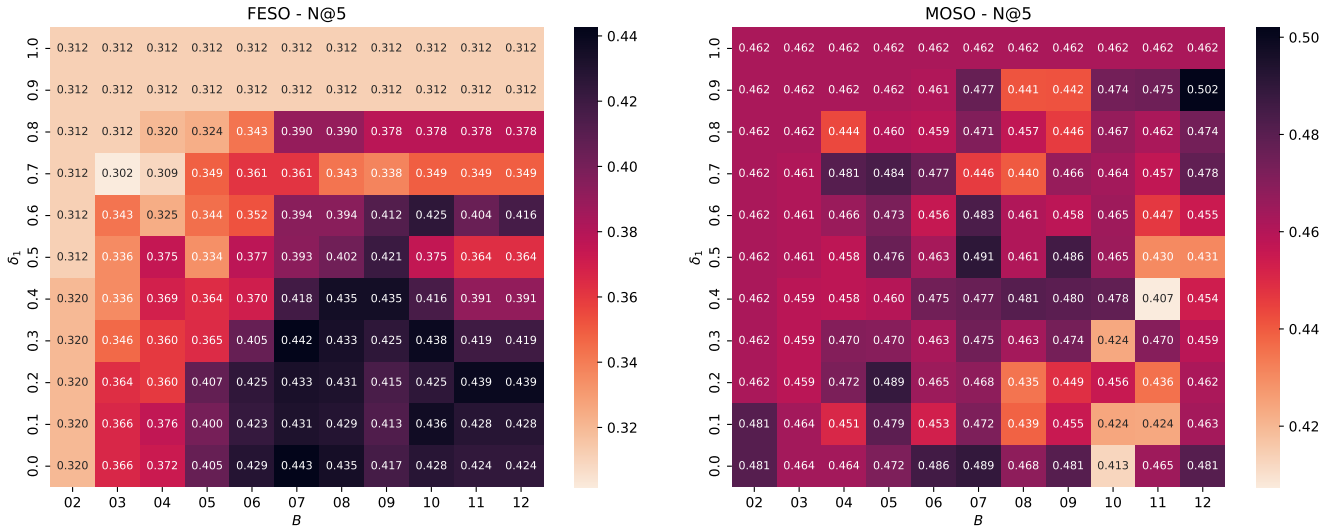

Figure 5. Hyper-parameter study of DIBRA-WIRE for the MOLO, MASO, FESO, and MOSO datasets. The heatmaps illustrate the fluctuation of N@5 for a variable number of buckets B (horizontal axis) and variable δ_1 values (in the range $[0, 1]$).

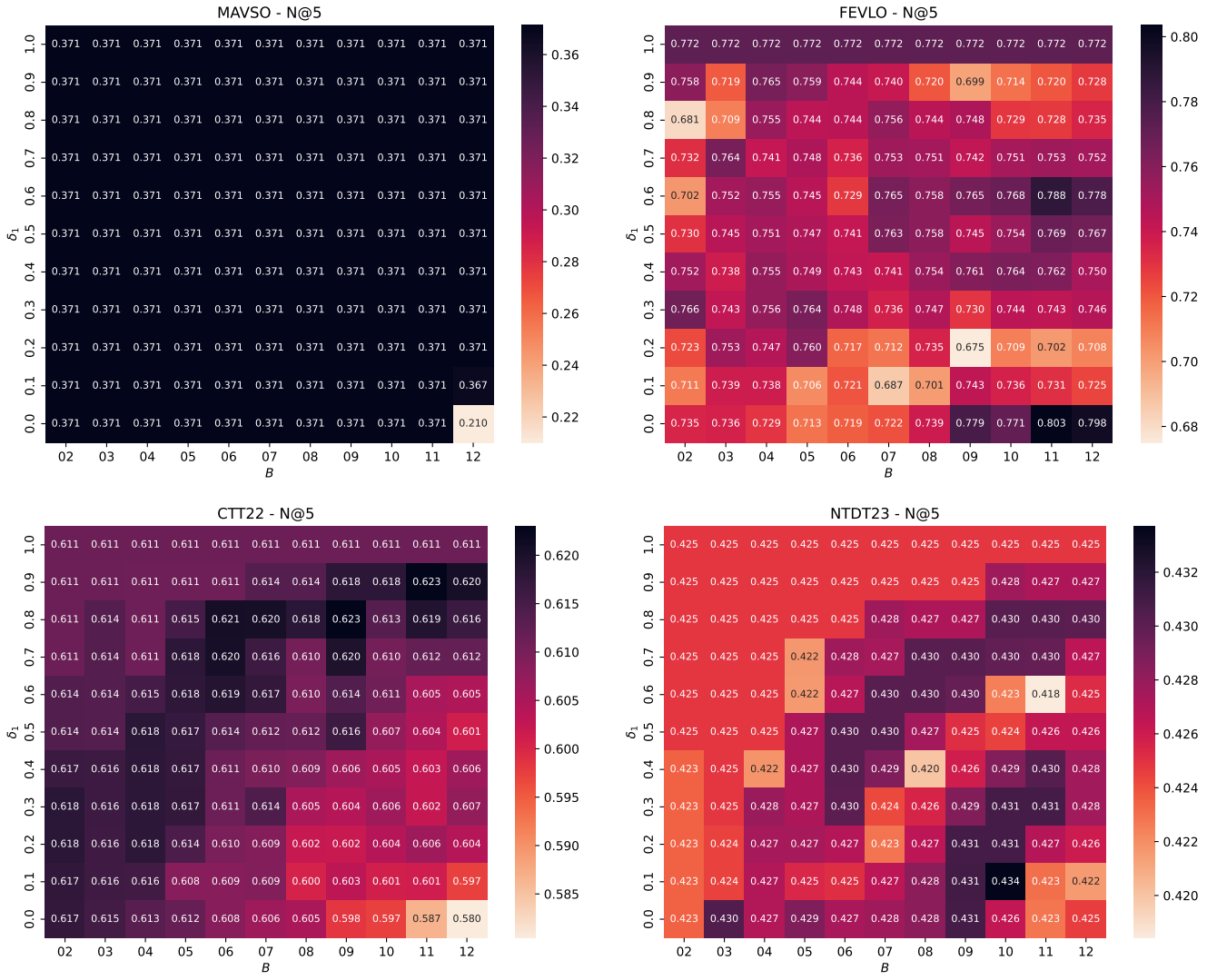

Figure 6. Hyper-parameter study of DIBRA-WIRE for the MAVSO, FEVLO, CTT22, and NTDT23 datasets. The heatmaps illustrate the fluctuation of N@5 for a variable number of buckets B (horizontal axis) and variable δ_1 values (in the range $[0, 1]$).

6. Conclusions and Future Work

In this paper, we introduced WIRE, a novel item selection approach for weighted rank aggregation applications. WIRE functions as a post-processing step to any weighted rank aggregation method and aims to further improve the quality of the output aggregate list. More specifically, our approach initially groups the input preference lists into a pre-defined number of buckets, according to the weights that have been assigned to their respective rankers by the original weighted aggregator. Based on that bucket, each preference list (that is, its respective ranker) is assigned a confidence score that quantifies its importance in a discretized manner.

Then, for each element of the aggregate list, WIRE computes a preservation score that derives from the sum of the confidence scores of the rankers who included it in their preference lists. In the sequel, the aggregate list is sorted in decreasing preservation score order. Consequently, WIRE favors the elements that have been selected by multiple expert rankers. The preservation score is designed to be free of manipulation; therefore, it does not depend on external factors like the individual rankings, pairwise wins and losses, or other score values that may have been assigned by the original aggregator. WIRE also employs the aforementioned confidence scores to compute a threshold value that determines the number of elements to be removed from the (sorted) aggregate list.

Our proposed method was theoretically analyzed and experimentally tested against a collection of 13 state-of-the-art rank aggregation methods in 8 benchmark datasets. The collection included both weighted and non-weighted aggregators, whereas the datasets were carefully selected in order to cover multiple diverse scenarios. The experiments highlighted the high retrieval effectiveness of WIRE in terms of Precision, normalized Discounted Cumulative Gain, and Mean Average Precision. Regarding future work, we intend to enhance WIRE by introducing more sophisticated discretization methods to determine the intervals (and ideally the number) of buckets where the rankers will be grouped. Examples of unsupervised binning methods include buckets of equal widths instead of equally sized buckets, or buckets that derive from the application of a clustering algorithm. We also plan to study alternative mathematical definitions for the confidence, preservation, and protection scores with the aim of improving the effectiveness of WIRE.

Author Contributions: Conceptualization, methodology, software, validation, formal analysis, resources, data curation, writing—original draft preparation, writing—review and editing: L.A. and P.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used in this study are publicly available on Kaggle: https://www.kaggle.com/datasets/lakritidis/rankaggregation. The implementation code of WIRE can be found in the open-source libraries FLAGR 1.0.20 & PyFLAGR 1.0.20: https://flagr.site, https://github.com/lakritidis/FLAGR, https://pypi.org/project/pyflagr/.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

WIRE	Weighted Item Removal
MAP	Mean Average Precision
nDCG	normalized Discounted Cumulative Gain

References

1. Chen, J.; Long, R.; Wang, X.l.; Liu, B.; Chou, K.C. PdRHP-PseRA: Detecting remote homology proteins using profile-based pseudo protein sequence and rank aggregation. *Sci. Rep.* **2016**, *6*, 32333.
2. Li, X.; Wang, X.; Xiao, G. A comparative study of rank aggregation methods for partial and top ranked lists in genomic applications. *Briefings Bioinform.* **2019**, *20*, 178–189. [CrossRef] [PubMed]
3. Gyarmati, L.; Orbán-Mihálykó, É.; Mihálykó, C.; Vathy-Fogarassy, Á. Aggregated Rankings of Top Leagues' Football Teams: Application and Comparison of Different Ranking Methods. *Appl. Sci.* **2023**, *13*, 4556. [CrossRef]
4. Oliveira, S.E.; Diniz, V.; Lacerda, A.; Merschmanm, L.; Pappa, G.L. Is rank aggregation effective in recommender systems? An experimental analysis. *ACM Trans. Intell. Syst. Technol. (TIST)* **2020**, *11*, 1–26. [CrossRef]
5. Bałchanowski, M.; Boryczka, U. A comparative study of rank aggregation methods in recommendation systems. *Entropy* **2023**, *25*, 132. [CrossRef] [PubMed]
6. Akritidis, L.; Katsaros, D.; Bozanis, P. Effective ranking fusion methods for personalized metasearch engines. In Proceedings of the 12th Panhellenic Conference on Informatics, Samos Island, Greece, 28–30 August 2008; pp. 39–43.
7. Wang, M.; Li, Q.; Lin, Y.; Zhou, B. A personalized result merging method for metasearch engine. In Proceedings of the 6th International Conference on Software and Computer Applications, Bangkok, Thailand, 26–28 February 2017; pp. 203–207.
8. Akritidis, L.; Katsaros, D.; Bozanis, P. Effective rank aggregation for metasearching. *J. Syst. Softw.* **2011**, *84*, 130–143. [CrossRef]
9. Bartholdi, J.; Tovey, C.A.; Trick, M.A. Voting schemes for which it can be difficult to tell who won the election. *Soc. Choice Welf.* **1989**, *6*, 157–165. [CrossRef]
10. Kilgour, D.M. Approval balloting for multi-winner elections. In *Handbook on Approval Voting*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 105–124.
11. Chen, D.; Xiao, Y.; Wu, J.; Pérez, I.J.; Herrera-Viedma, E. A Robust Rank Aggregation Framework for Collusive Disturbance Based on Community Detection. *Inf. Process. Manag.* **2025**, *62*, 104096. [CrossRef]
12. Ma, K.; Xu, Q.; Zeng, J.; Liu, W.; Cao, X.; Sun, Y.; Huang, Q. Sequential Manipulation Against Rank Aggregation: Theory and Algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 9353–9370. [CrossRef] [PubMed]
13. Akritidis, L.; Fevgas, A.; Bozanis, P.; Manolopoulos, Y. An unsupervised distance-based model for weighted rank aggregation with list pruning. *Expert Syst. Appl.* **2022**, *202*, 117435. [CrossRef]
14. de Borda, J.C. Mémoire sur les élections au scrutin. In *Histoire de l'Academie Royale des Sciences*; Imprimerie Royale: Paris, France, 1781; pp. 657–665.
15. De Condorcet, N. *Essai sur l'Application de l'Analyse à la Probabilité des Décisions Rendues à la Pluralité des Voix*; Imprimerie Royale: Paris, France, 1785.
16. Emerson, P. The original Borda Count and partial voting. *Soc. Choice Welf.* **2013**, *40*, 353–358. [CrossRef]
17. Montague, M.; Aslam, J.A. Condorcet fusion for improved retrieval. In Proceedings of the 11th ACM International Conference on Information and Knowledge Management, McLean, VA, USA, 4–9 November 2002; pp. 538–548.
18. Li, G.; Xiao, Y.; Wu, J. Rank Aggregation with Limited Information Based on Link Prediction. *Inf. Process. Manag.* **2024**, *61*, 103860. [CrossRef]
19. Dwork, C.; Kumar, R.; Naor, M.; Sivakumar, D. Rank aggregation methods for the Web. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 613–622.
20. DeConde, R.P.; Hawley, S.; Falcon, S.; Clegg, N.; Knudsen, B.; Etzioni, R. Combining results of microarray experiments: A rank aggregation approach. *Stat. Appl. Genet. Mol. Biol.* **2006**, *5*, 15. [CrossRef] [PubMed]
21. Kolde, R.; Laur, S.; Adler, P.; Vilo, J. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics* **2012**, *28*, 573–580. [CrossRef] [PubMed]
22. Kemeny, J.G. Mathematics without numbers. *Daedalus* **1959**, *88*, 577–591.
23. Young, H.P.; Levenglick, A. A consistent extension of Condorcet's election principle. *SIAM J. Appl. Math.* **1978**, *35*, 285–300. [CrossRef]
24. Farah, M.; Vanderpooten, D. An outranking approach for rank aggregation in information retrieval. In Proceedings of the 30th ACM Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, 23–27 July 2007; pp. 591–598.
25. Pihur, V.; Datta, S.; Datta, S. Weighted rank aggregation of cluster validation measures: A Monte Carlo cross-entropy approach. *Bioinformatics* **2007**, *23*, 1607–1615. [CrossRef] [PubMed]
26. Desarkar, M.S.; Sarkar, S.; Mitra, P. Preference relations based unsupervised rank aggregation for metasearch. *Expert Syst. Appl.* **2016**, *49*, 86–98. [CrossRef]
27. Chatterjee, S.; Mukhopadhyay, A.; Bhattacharyya, M. A weighted rank aggregation approach towards crowd opinion analysis. *Knowl.-Based Syst.* **2018**, *149*, 47–60. [CrossRef]
28. Akritidis, L.; Alamaniotis, M.; Bozanis, P. FLAGR: A flexible high-performance library for rank aggregation. *SoftwareX* **2023**, *21*, 101319. [CrossRef]

29. Renda, M.E.; Straccia, U. Web metasearch: Rank vs. Score based rank aggregation methods. In Proceedings of the 2003 ACM Symposium on Applied Computing, Melbourne, FL, USA, 9–12 March 2003; pp. 841–846.
30. Copeland, A.H. *A Reasonable Social Welfare Function*; Technical Report; University of Michigan: Ann Arbor, MI, USA, 1951.