Sequence-wise Kendall's Tau Distance: Aggregate Rankings Meaningfully

Wangze Ni †, Peng Cheng *, Lei Chen †
†*The Hong Kong University of Science and Technology, Hong Kong, China*
{wniab, leichen}@cse.ust.hk
**East China Normal University, Shanghai, China*
pcheng@sei.ecnu.edu.cn

***Abstract*—Rank aggregation is a classical problem that aims at generating a consensus ranking over a set of items from a collection of preferences, namely base rankings, over some of the items. Kendall's Tau distance (KTD) is a classical and widely implemented metric to measure the difference between two rankings, which counts the number of item pairs having different orders in the two rankings. However, in some scenarios such as transaction ordering, the ranking is meaningful, only if the entire transaction sequence is maintained. Since KTD only considers the order of each pair, KTD-based solutions will break transaction sequences into transaction pairs and reorganize them in the consensus ranking. Then, the obtained consensus ranking may be meaningless.**

Therefore, in this paper, we propose a novel metric, the sequence-wise Kendall's Tau distance (SWKTD), and analyze its properties. If two items having different related orders in two rankings, the SWKTD of these two rankings is 1; otherwise, it is 0. Then, we formally define the sequence-wise rank aggregation (SWRA) problem. Given a set of base rankings, the SWRA problem aims to generate a consensus ranking having a minimal sum of SWKTDs with the base rankings. We prove the SWRA problem is NP-hard. Therefore, we propose two approximation algorithms, namely the CrossMerge algorithm and the BubbleRank algorithm. We evaluate the performance of our solutions over real and synthetic data sets.

***Index Terms*—Rank Aggregation, Distance Metric**

I. Introduction

Rank aggregation is a classical and practical problem, whose study has a long history, tracing back to the works in the social choice field [1], [2] in the 18th century. Recent interests in artificial intelligence [3], [4] and databases [5], [6], [7], [8] have renewed the developments of rank aggregation methods and distance metrics. Specifically, given an item universe and a set of base rankings where each one orders a subset of the item universe, the rank aggregation problem aims to generate a consensus ranking over the items having a minimal difference with the base rankings. Kendall's Tau distance [9] (KTD) is a widely used metric to measure the difference between two rankings, which counts the number of item pairs having different orders in the two rankings.

However, in some scenarios such as transaction ordering, a base ranking has some specific meanings and only the entire sequence is maintained, the meaning can be achieved. For example, in exchanges with continuous-limit order books [10], the final trading price depends on the order of transactions. Besides, in decentralized systems, where transactions may have dependencies (or conflicts) [11], [12], different orders of transactions may lead to different execution results. Replicates/nodes may give diverse priorities to transactions for some purposes or criteria [13]. When merging these preferences into a consensus transaction sequence, KTD-based methods will break sequences into pairs and reorganize these pairs. Then, the obtained consensus ranking may be meaningless and satisfy only a few purposes/criteria. Here is an example.

Example 1 (Motivation Example)**.** *In a blockchain system, three transactions are proposed and should be ordered in a block. Specifically, t_1 increases the balance of account B to that of account A, t_2 increases the balance of A by one thousand, and t_3 deducts twenty percent from both accounts.*

Nine Replicates/nodes orders the transactions by their purposes/criteria, and propose three different rankings. Three base rankings are $r_1 = \langle t_1 \succ t_2 \succ t_3 \rangle$, where $t_i \succ t_j$ means t_i is executed before t_j. The purpose of r_1 is to let the sum of accounts' balance be minimal. Four base rankings are $r_2 = \langle t_3 \succ t_2 \succ t_1 \rangle$, whose purposes are to maximize the sum of accounts' balance. Two base rankings are $r_3 = \langle t_1 \succ t_2 \rangle$, whose purposes are to let the balances of the accounts be different.

Table I demonstrates the KTDs between each possible consensus ranking with base rankings. As defined in [14], if it is not the case that both t_i and t_j appear in both ranking r_h and r_g, then the pair (t_i, t_j) contributes nothing to the KTD between r_h and r_g. For example, since t_3 does not appear in r_3 and the orders of (t_1, t_2) pair in r_3 and r_8 are the same, the KTD between r_3 and r_8 is 0.

In Table I, r_8 is the consensus ranking having a minimum sum of KTDs with base rankings. According to r_8, although the balances of the accounts are different, the sum of the accounts' balance is neither minimized nor maximized. In other words, r_8 only satisfies two base rankings' purposes (i.e., two r_3). However, r_4 satisfies five base rankings' purposes (i.e., three r_1 and two r_3).

Motivated by Example 1, in this paper, we first propose a novel metric, sequence-wise Kendall's Tau distance (SWKTD), and analyze its properties. Specifically, if two items having different related orders in two rankings, the SWKTD between the two rankings is 1; otherwise, it is 0. Then, we formally define the sequence-wise rank aggregation

TABLE I
MOTIVATION EXAMPLE.

ranking	KTD
$r_4 = \langle t_1 \succ t_2 \succ t_3 \rangle$	12
$r_5 = \langle t_1 \succ t_3 \succ t_2 \rangle$	11
$r_6 = \langle t_2 \succ t_1 \succ t_3 \rangle$	13
$r_7 = \langle t_2 \succ t_3 \succ t_1 \rangle$	12
$r_8 = \langle t_3 \succ t_1 \succ t_2 \rangle$	10
$r_9 = \langle t_3 \succ t_2 \succ t_1 \rangle$	11

(SWRA) problem. Given a set of base rankings, the SWRA problem aims to generate a consensus ranking a minimal sum of SWKTDs with the base rankings. We prove the SWRA problem is NP-hard, which is intractable to retrieve an exact solution within polynomial time. Therefore, we propose two approximation algorithms, namely the CrossMerge algorithm and the BubbleRank algorithm. The CrossMerge algorithm repeatedly merges a base ranking to obtain an updated consensus ranking with a minimum SWKTD. The BubbleRank repeatedly adds an item to the end of the current consensus ranking with a minimum SWKTD.

To the best of our knowledge, this is the first study proposing a sequence-wise Kendall's Tau distance and proposing solutions to retrieve a consensus ranking with a minimum SWKTD. Our work provides new insights on how to merge base rankings to retrieve a good consensus ranking. In summary, we have made the following contributions:

- In Section II, we define a novel metric, sequence-wise Kendall's Tau distance (SWKTD), analyze the properties of SWKTD, formulate the sequence-wise rank aggregation (SWRA) problem, and prove the NP-hardness of the SWRA problem.
- We propose an effective approximation algorithm, namely CrossMerge, in Section III.
- We propose an efficient approximation algorithm, namely BubbleRank, in Section IV.
- In Section **??**, we make a comprehensive experimental study to test our solutions' performances over real and synthetic data sets.

In addition, we discuss the related works in Section **??** and conclude our work in Section **??**.

II. PROBLEM DEFINITION

In this section, we first formally define the concept of ranking. Then, we propose our novel concept, sequence-wise Kendall's tau distance (SWKTD), and theoretically analyze its properties. After that, we formulate the sequence-wise rank aggregation (SWRA) problem. We prove the SWRA problem is NP-hard.

A. A Ranking

In practice, users may be only interested in a subset of the item universe, and it is hard to rank all items [15]. For instance, in Example 1, the proposers of r_3 only care the order of t_1 and t_2. Therefore, in this paper, we define a ranking by the partial rank model ranking [16], where a ranking can only order a subset of the item universe T. Note that a ranking in the partial rank model also can order all items in the item universe.

Definition 1 (A ranking)**.** A ranking, denoted by $r_i = \langle \phi_{i,1} \succ \phi_{i,2} \succ \cdots \succ \phi_{i,|r_i|} \rangle$, is an ordered sequence over $|r_i|$ items, where $|r_i|$ is the number of transactions ordered in r_i, $\phi_{i,j}$ is the item having j^{th} priority in r_i, and $\forall \phi_{i,j} \in r_i$, $\phi_{i,j} \in T$.

In the scenario of transaction ordering, there is no case that two transactions have to be executed simultaneously. Thus, in this paper, we do not consider the tie case where two transactions have the same priority. When $|r_i| = |T|$, where $|T|$ is the number of transactions in the transaction universe T, the ranking r_i is a *full ranking* (i.e., ordering all items in the item universe). A consensus ranking should be a full ranking. In this paper, we use $t_i \in r_j$ to indicate that a ranking r_j orders a transaction t_i in the sequence. We use $t_i \notin r_j$ to indicate that a ranking r_j does not order a transaction t_i in the sequence. For instance, in Example 1, $\phi_{4,1} = t_1$, $n_3 = 2$, $t_3 \in r_1$, but $t_3 \notin r_3$.

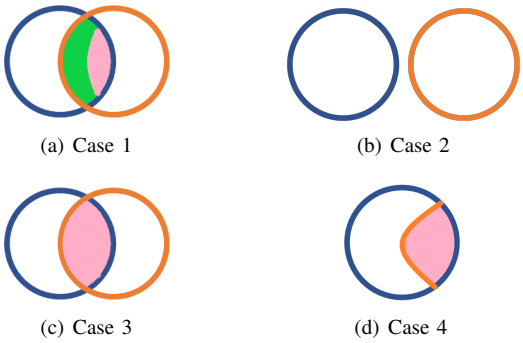
B. The Sequence-wise Kendall's Tau Distance

To measure the distance between two rankings, in this subsection, we formally define a novel concept, the sequence-wise Kendall's Tau distance (SWKTD).

Definition 2 (The sequence-wise Kendall's Tau distance)**.** The sequence-wise Kendall's Tau distance (SWKTD) between two rankings r_i and r_j is denoted by $\delta(r_i, r_j)$. If two transactions contained in the two rankings whose related orders in the two rankings are different, the distance is 1 (i.e, $\delta(r_i, r_j)$ = 1); otherwise, the distance is zero (i.e., $\delta(r_i, r_j)$ = 0).

We use $r_i \overline{\wedge} r_j = r'$ to indicate that there exists a transaction sequence r' that is maintained in both r_i and r_j. We termed r' as the *intersection sequence* of r_i and r_j. We use $r_i \cap r_j = T'$ to indicate that there exists a set of transactions ordered in both r_i and r_j (but they may have different orders). We termed r' as the *intersection set* of r_i and r_j. We use $r_i \subset r_j$ to indicate that the transaction sequence of r_i is maintained in r_j, termed as r_i is the subset of r_j. r' and T can be empty. For instance, in Example 1, $r_1 \overline{\wedge} r_2 = \emptyset$, $r_1 \cap r_2 = \{t_1, t_2, t_3\}$, and $r_3 \subset r_1$.

As shown in Figure 1, for two rankings, there are four possible cases. For case 1, shown in Figure 1(a), several transactions are both ordered in the two rankings, but only a subset of the transactions maintain the same order sequence in the two rankings, i.e., $|r'| > |T'|$. Thus, by Definition 2, $\delta(r_i, r_j) = 1$. For case 2, shown in Figure 1(b), no transaction is ordered in the two rankings simultaneously, i.e., $|r'| = 0$. Thus, by Definition 2, $\delta(r_i, r_j) = 0$. For case 3, shown in Figure 1(c), several transactions are both ordered in the two rankings, and the sequences of these transactions in the two rankings are the same, i.e., $|r'| = |T'|$. Thus, by Definition 2, $\delta(r_i, r_j) = 0$. For case 4, shown in Figure 1(d), the transaction sequence of one ranking is maintained in another transaction, i.e., $|T'| = |r_i|$. Thus, $r_i \subset r_j$, and by Definition 2, $\delta(r_i, r_j) = 0$.



(a) Case 1 (b) Case 2

(c) Case 3 (d) Case 4

Fig. 1. An Example of Mixing Services.

Next, by convention [17], we theoretically analyze the properties of the sequence-wise Kendall's Tau distance. We prove that SWKTD has non-negativity and symmetry. Specifically, the non-negativity requires that the SWKTD between any two rankings cannot be negative, and the symmetry requires that the SWKTD from a ranking r_i to a ranking r_j should be equal to the SWKTD from r_j to r_i.

Theorem II.1. *The sequence-wise Kendall's Tau distance satisfies non-negativity and symmetry.*

Proof. As defined in Definition 2, for any two rankings r_i and r_j, the sequence-wise Kendall's Tau distance $\delta(r_i, r_j) \geq 0$. Thus, the sequence-wise Kendall's Tau distance satisfies non-negativity.

As defined in Definition 2, for any two rankings r_i and r_j, $\delta(r_i, r_j) = \delta(r_j, r_i)$. Thus, the sequence-wise Kendall's Tau distance satisfies symmetry. □

Another common property is the identity of indiscernibles [17], which requires that if $\delta(r_i, r_j) = 0$, r_i and r_j should be the same. However, the sequence-wise Kendall's Tau distance does not satisfy the identity of indiscernibles. For example, when $r_1 = \langle t_1 \succ t_2 \rangle$ and $r_2 = \langle t_3 \succ t_4 \rangle$, $\delta(r_1, r_2) = 0$ but r_1 and r_2 are totally different. In Subsection III-A, we propose a variant of the identity of indiscernibles, the compatibility. We prove that the sequence-wise Kendall's Tau distance satisfies the incompatibility. Inspired by the incompatibility, we design the CrossMerge algorithm.

Another critical property is the triangle inequality [17]. Specifically, the triangle inequality requires that $\forall r_i, r_j, r_k$, $\delta(r_i, r_k) + \delta(r_j, r_k) \geq \delta(r_i, r_j)$. Unfortunately, the sequence-wise Kendall's Tau distance does not satisfy the triangle inequality. For example, when $r_1 = \langle t_1 \succ t_2 \rangle$, $r_2 = \langle t_3 \succ t_4 \rangle$, and $r_3 = \langle t_2 \succ t_1 \rangle$, $\delta(r_1, r_2) + \delta(r_3, r_2) = 0 < \delta(r_1, r_3) = 1$. In [18], researchers have stated that the cause that a distance metric does not have the triangle inequality is not always the poor design. Many famous distance metrics do not have the triangle inequality either, like Longest Common Subsequences [19], DTW [20], and Edit Distance on Real sequence [21]. Thus, although SWKTD does not have the triangle inequality, it still is reasoning. In addition, in Subsection III-C, we propose a variant of the triangle inequality, the full-ranking triangle inequality. We prove that the sequence-wise Kendall's Tau distance satisfies the full-ranking triangle inequality. Utilizing this property, we calculate the lower bound of the optimal answer for the SWRA problem.

C. The Sequence-wise Rank Aggregation Problem

Based on the concepts of ranking and SWKTD, in this subsection, we formally define the sequence-wise rank aggregation (SWRA) problem.

Definition 3 (The sequence-wise rank aggregation problem)**.** Given a set of base rankings R, the sequence-wise rank aggregation (SWRA) problem aims to find a consensus ranking π having a minimum sum SWKTDs with the base rankings, i.e., $D(\pi, R) = \sum_{r_i \in R} \delta(\pi, r_i)$ is minimized.

Suppose in Example 1, $R = \{r_1, r_2, r_3\}$. Then, $D(r_4, R) = 4$, $D(r_5, R) = 7$, $D(r_6, R) = 9$, $D(r_7, R) = 9$, $D(r_8, R) = 7$, and $D(r_9, R) = 5$. Thus, r_4 is the consensus ranking having a minimum sum of SWKTDs with rankings in R, which fits our intuition.

In addition, we prove that the result of the SWRA problem satisfies the Condorcet consistency property [22]. The Condorcet consistency property states that if a transaction has the highest priority in each base ranking, termed as a Condorcet winner, it can be ranked with the highest priority in the consensus ranking.

Theorem II.2. *The result of the SWRA problem satisfies the Condorcet consistency property.*

Proof. Suppose t^* is the transaction having the highest priority in each base ranking, R is the set of base rankings, and π^* is the consensus ranking where t^* has the highest priority. Thus, $D(\pi^*, R) \leq |R|$, where $|R|$ is the number of base rankings in R. Assume π' is a consensus ranking where t^* does not have the highest priority. Since t^* has the highest priority in each base ranking and does not have the highest priority in π', the SWKTD between π' and each base ranking is 1. Thus, $D(\pi', R) = |R| \geq D(\pi^*, R)$. In other words, no other consensus ranking has a smaller SWKTD sum than π^*. Thus, π^* can be the consensus ranking of the SWRA problem, and t^* can have the highest priority, which completes our proof. □

For example, there are three base rankings, $r_1 = \langle t_1 \succ t_2 \succ t_3 \rangle$, $r_2 = \langle t_1 \succ t_3 \succ t_2 \rangle$, and $r_3 = \langle t_1 \succ t_3 \rangle$. Then, it must be correct that to rank t_1 in the consensus ranking with the highest priority. Theorem II.2 shows that the SWRA problem is reasoning.

D. NP-hardness

However, it is difficult to find an optimal solution to the sequence-wise rank aggregation problem. In this subsection, we formally prove that the SWRA problem is NP-hardness by reducing it from the traditional KTD-based rank aggregation problem [23]. Suppose $\mathbb{I}_x = 1$ if the condition x is true; otherwise, $\mathbb{I}_x = 0$.

TABLE II
SYMBOLS AND DESCRIPTIONS.

Symbol	Description
T	the transaction universe
t_i	a transaction
$t_i \succ t_j$	t_i has a higher priority than t_j
R	a set of base rankings
r_i	a ranking
$\phi_{i,j}$	the transaction having j^{th} priority in r_i
$t_i \in r_j$	t_i is ordered in r_j
$t_i \notin r_j$	t_i is not ordered in r_j
$r_i \overline{\wedge} r_j$	the sequence is maintained in both r_i and r_j
$r_i \cap r_j$	the transactions that are ordered in both r_i and r_j
$r_i \subset r_j$	the transaction sequence of r_i is maintained in r_j
$\delta(r_i, r_j)$	the SWKTD between r_i and r_j
π	the consensus ranking
$D(\pi, R)$	the SWKTDs between π and base rankings

Theorem II.3. *The sequence-wise rank aggregation problem is NP-hard.*

Proof. We prove that the SWRA problem is NP-hard by reducing it from the traditional KTD-based rank aggregation problem [23]. The KTD between two rankings r_i and r_j, denoted by $\delta_p(r_i, r_j)$, is the number of transaction pairs having different orders in r_i and r_j, i.e., $\delta_p(r_i, r_j) = \sum_{t_h \in r_i \cap r_j} \sum_{t_g \in r_i \cap r_j} |\mathbb{I}_{L(r_i,t_h)>L(r_j,t_h)} - \mathbb{I}_{L(r_i,t_g)>L(r_j,t_g)}|$. Then, we can describe an instance of the KTD-based rank aggregation problem as follows: Given a set of base rankings K, the KTD-based rank aggregation problem aims to find a consensus ranking c having a minimum sum of KTDs, denoted by $KTD(c, K)$, with base rankings in K, where $KTD(c, K) = \sum_{r_i \in K} \delta_p(r_i, c)$.

Given a KTD-based rank aggregation problem instance, we can construct an instance of the SWRA problem accordingly: For each transaction pair in each ranking in K, generate a base ranking. The set of these base rankings is R. Then, we can see that to minimize $D(\pi, R)$ is equal to minimize $KTD(c, K)$.

Thus, we can solve the KTD-based rank aggregation problem if we can solve the transformed SWRA problem. Since researchers have proved that the KTD-based rank aggregation is NP-hard [23], the SWRA problem is also NP-hard. Thus, Theorem II.3 is proved. □

Thus, it is intractable to accurately solve the SWRA problem within polynomial time. Thus, we turn to design approximation algorithms to retrieve the near-optimal answer for the SWRA problem within polynomial time. We propose two approximation algorithms in Section III and Section IV. Table II summarizes the notations of common symbols in this paper.

III. THE CROSSMERGE ALGOTIHM

In this section, we first introduce the base idea of the CorssMerge algorithm. Then, we describe the details of the algorithm and demonstrate a running example. Finally, we theoretically analyze the performance of the algorithm.

A. Basic Idea

We term two rankings r_i and r_j in case 2, case 3, or case 4 in Figure 1 as two compatible rankings. Thus, for two compatible rankings, their SWKTD must be zero, i.e., $\delta(r_i, r_j) = 0$. Besides, we term the transactions in their intersection sequence (i.e., $r_i \overline{\wedge} r_j$) as *anchor transactions*. We use $A(i, j)$ to indicate the j^{th} anchor transaction in r_i. We use $s(t_i, t_j, r_k)$ to indicate the sequence in r_k between t_i and t_j. In particular, $s(null, t_j, r_k)$ indicates the sequence in r_k before t_j, and $s(t_i, null, r_k)$ indicates the sequence in r_k after t_i. For instance, in Example 1, r_3 and r_5 are two compatible rankings, t_1 and t_2 are two anchor transactions in r_5, $A(5, 2) = t_2$, $s(null, t_3, r_5) = \langle t_1 \rangle$, and $s(t_1, null, r_5) = \langle t_4 \succ t_2 \rangle$. Then, we find that, given two compatible rankings, we can merge them to obtain a ranking having zero SWKTDs with them.

Theorem III.1. *Suppose r_i and r_j are two compatible rankings having $n_{i,j}$ anchor transactions. We construct a new ranking r as follows: (1) set $r = r_j$ and add $s(null, A(i, 1), r_i)$ before $A(i, 1)$ in r; (2) $\forall k \in [1, n_{i,j})$, add $s(A(i, k), A(i, k+1), r_i)$ before $A(i, k+1)$ in r; and (3) add $s(A(i, n_{i,j}), null, r_i))$ to the end of r (when $n_{i,j} = 0$, add r_i to the end of r). The SWKTD from r to r_i and r_j are both 0, i.e., $\delta(r_i, r) = 0, \delta(r_j, r) = 0$.*

Proof. Since initially in step (1), $r = r_j$, and the added transactions in the last two steps do not change the related orders of transactions in r_j, by Definition 2, $\delta(r_j, r) = 0$. The related order of any two anchor transactions is the same in r and r_i. Besides, since the transactions between any two adjacent anchor transactions in r_i are added to r as their original order sequence, by Definition 2, $\delta(r_i, r) = 0$. Thus, the theorem is proved. □

For example, when $r_i = \langle t_1 \succ t_2 \succ t_3 \succ t_4 \rangle$ and $r_j = \langle t_5 \succ t_1 \succ t_6 \succ t_3 \rangle$, as the construction way in Theorem III.1, we obtain $r = \langle t_5 \succ t_1 \succ t_6 \succ t_2 \succ t_3 \succ t_4 \rangle$. By Definition 2, $\delta(r_i, r) = 0$ and $\delta(r_j, r) = 0$. Thus, by Theorem III.1, we can initialize the consensus ranking as empty and repeatedly merge compatible rankings in R into the consensus ranking until it becomes a full ranking.

Next, we propose a variant of the identity of indiscernibles, namely incompatibility. We term two rankings r_i and r_j in case 1 in Figure 1 as two incompatible rankings. Thus, for two incompatible rankings, their SWKTD must be one, i.e., $\delta(r_i, r_j) = 1$.

Definition 4 (Incompatibility)**.** If r_i and r_j are two incompatible rankings, no ranking r_k that contains all transactions in $r_i \cup r_j$ and has zero SWKTDs with r_i and r_j.

We prove that the SWKTD satisfies the incompatibility.

Theorem III.2. *The sequence-wise Kendall's Tau distance satisfies the incompatibility.*

Proof. Suppose r_i and r_j are two incompatible rankings. $T^{\#}$ is the set of transactions contained in rankings, i.e., $T^{\#} = \{t_k | t_k \in r_i \text{ or } t_k \in r_j\}$. Since r_i and r_j are incompatible,

there are two transactions having different related orders in r_i and r_j, denoted as t_g and t_h. Assume the related order of t_g and t_h in r_i is $s_i^{\#} = \langle t_g \succ t_h \rangle$. Then, the related order of t_g and t_h in r_j is $s_j^{\#} = \langle t_h \succ t_g \rangle$.

We prove the theorem by contradiction. Assume the sequence-wise Kendall's Tau distance does not satisfy the incompatibility. Thus, there is a ranking, denoted as r_k, ordering all transactions in $T^{\#}$ and $\delta(r_k, r_i) = \delta(r_k, r_j) = 0$. Suppose the related order of t_g and t_h in r_k is $s_k^{\#}$. Since $\delta(r_k, r_i) = 0$, $s_k^{\#} = s_i^{\#}$. Similarly, since $\delta(r_k, r_j) = 0$, $s_k^{\#} = s_j^{\#}$, which conflicts with the fact that $s_i^{\#} \neq s_j^{\#}$. Thus, the assumption does not hold, and the sequence-wise Kendall's Tau distance satisfies the incompatibility. □

Thus, if we find a base ranking r_i in R is incompatible with the current consensus ranking π, whatever later we merge which base rankings into the consensus ranking, $\delta(r_i, \pi) = 1$. In addition, if we want to obtain a consensus ranking with a minimum $D(\pi, R)$, after merging a base ranking into π, we should let the number of π' incompatible rankings in R be as small as possible.

B. Algorithm Description

Based on the basic ideas in Subsection III-A, we design the CrossMerge algorithm, which merges base rankings in R into the consensus ranking as the way in Theorem III.1 in a greedy manner.

Algorithm 1 illustrates the pseudo-code of the CrossMerge algorithm. We first initialize the consensus ranking π as empty and initialize R' as R (line 1). R' will be used to store the base rankings that do not violate the updated consensus ranking π. Then, we merge base rankings in R' into the base ranking π in a greedy manner (line 2-9). Specifically, for each base ranking r_i in R', we merge it into π as the way in Theorem III.1, and the obtained ranking is denoted by π_i (line 4). Then, we calculate the sum of SWKTDs, $D(\pi_i, R)$, between π_i and the base rankings in R, denoted by d_i (line 5). After that, we update the consensus ranking as the ranking π_i with the lowest d_i (line 6). Then, we update R' (line 7-9). Specifically, if a base ranking is the subset of π, the transactions in it are all ordered in π. Thus, we can safely remove it from R'. Besides, if a base ranking r_i is incompatible with the current consensus ranking π (i.e., $\delta(r_i, \pi) = 1$), by Theorem III.2, r_i cannot be merged into π. Thus, we can safely remove it from R', too. When the while-loop terminates, some transactions may not be ordered (line 11). Since the base rankings ordering these transactions must be incompatible with π over some other transactions, the priorities of these transactions will not affect $D(\pi, R)$. Thus, we directly add these transactions to the end of π (line 12). Finally, we return π as the consensus ranking (line 13). Here is a running example of the CrossMerge algorithm.

Example 2 (Running Example)**.** *There are seven base rankings, $r_1 = \langle t_4 \succ t_1 \succ t_2 \rangle$, $r_2 = \langle t_1 \succ t_4 \succ t_2 \rangle$, $r_3 = \langle t_4 \succ t_3 \succ t_2 \rangle$, $r_4 = \langle t_2 \succ t_4 \rangle$, $r_5 = \langle t_3 \succ t_2 \succ t_4 \rangle$, $r_6 = \langle t_4 \succ t_3 \rangle$, and $r_7 = \langle t_2 \succ t_3 \succ t_4 \rangle$. The transaction universe is $T = \{t_1, t_2, t_3, t_4\}$.*

Algorithm 1: The CrossMerge algorithm.

Input: a transaction universe T and a set of base rankings R
Output: a consensus ranking π

```
1  π = ∅, R' = R;
2  while R' ≠ ∅ do
3      foreach r_i ∈ R' do
4          π_i ← merge r_i into π like Theorem III.1;
5          d_i = D(π_i, R);
6      π ← π_i with the lowest d_i;
7      foreach r_i ∈ R' do
8          if δ(r_i, π) == 1 || r_i ⊂ π then
9              remove r_i from R';
10 foreach t_i ∈ T do
11     if t_i ∉ π then
12         add t_i to the end of π;
13 return π;
```

When we run the CrossMerge algorithm to solve this SWRA problem instance, initially, $\pi = \emptyset$ and $R' = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$. Then, in the first round of the while-loop, $\forall i \in [1, 7], \pi_i = r_i$. Therefore, $d_1 = 4$, $d_2 = 4$, $d_3 = 3$, $d_4 = 3$, $d_5 = 5$, $d_6 = 2$, and $d_7 = 5$. Therefore, we update π as $\pi_6 = \langle t_4 \succ t_3 \rangle$. Besides, since $r_6 \subset \pi$ and $\delta(\pi, r_5) = \delta(\pi, r_7) = 1$, we update R' as $\{r_1, r_2, r_3, r_4\}$. Next, in the second round of the while-loop, $\pi_1 = \langle t_4 \succ t_3 \succ t_1 \succ t_2 \rangle$, $\pi_2 = \langle t_1 \succ t_4 \succ t_3 \succ t_2 \rangle$, $\pi_3 = \langle t_4 \succ t_3 \succ t_2 \rangle$, and $\pi_4 = \langle t_2 \succ t_4 \succ t_3 \rangle$ Therefore, $d_1 = 4$, $d_2 = 4$, $d_3 = 3$, and $d_4 = 5$. Therefore, we update π as $\pi_3 = \langle t_4 \succ t_3 \succ t_2 \rangle$. Besides, since $r_3 \subset \pi$ and $\delta(r_4, \pi) = 1$, we update R' as $\{r_1, r_2\}$. Next, in the third round of the while-loop, $\pi_1 = \langle t_4 \succ t_3 \succ t_1 \succ t_2 \rangle$, and $\pi_2 = \langle t_1 \succ t_4 \succ t_3 \succ t_2 \rangle$. Therefore, $d_1 = d_2 = 4$. Therefore, we update π as $\pi_1 = \langle t_4 \succ t_3 \succ t_1 \succ t_2 \rangle$. Besides, since $\delta(r_2, \pi) = 1$, we update R' as an empty set. Therefore, the while-loop terminates, and the consensus ranking returned by the CrossMerge algorithm is $\langle t_4 \succ t_3 \succ t_1 \succ t_2 \rangle$, whose $D(\pi, R)$ is 4.

C. Theoretical Analysis

In this section, we theoretically analyze the approximation ratio and the time complexity of the CrossMerge algorithm.

As we analyzed in Subsection II-B, SWKTD does not satisfy the triangle inequality, which usually is utilized to calculate the bound of distances. For the distance metrics without triangle inequality, a common way is to find a correlative and reasoning alternative with triangle inequality [24], [25]. To prove the approximation ratio, we first propose a variant of the triangle inequality, namely full-ranking triangle inequality.

Definition 5 (Full-ranking triangle inequality)**.** If for any two rankings, r_i and r_j, and a full ranking r_k, it satisfies that

$\delta(r_i, r_j) \leq \delta(r_i, r_k) + \delta(r_j, r_k)$, the distance metric $\delta(\cdot, \cdot)$ satisfies the full-ranking triangle inequality.

We prove that our SWKTD has the full-ranking triangle inequality.

Theorem III.3. *The SWKTD satisfies the full-ranking triangle inequality.*

Proof. Suppose r_i and r_j are two rankings, and r_k is a full ranking. By Theorem II.1, $\delta(t_i, t_k) \geq 0$ and $\delta(t_j, t_k) \geq 0$. Therefore, when $\delta(t_i, t_j) = 0$, it must be correct that $\delta(r_i, r_j) \leq \delta(r_i, r_k) + \delta(r_j, r_k)$.

As analyzed in Subsection II-B, when $\delta(r_i, r_j) = 1$, it must be case 1 in Figure 1(a). Thus, there exist two transactions, $\bar{t}$ and $\hat{t}$, having different related orders in r_i and r_j. Since r_k is a full ranking, r_k also orders $\bar{t}$ and $\hat{t}$. Suppose the sequence of $\bar{t}$ and $\hat{t}$ in r_i, r_j, and r_k is $\widetilde{s_i}$, $\widetilde{s_j}$, and $\widetilde{s_k}$, respectively. Thus, $\widetilde{s_i} \neq \widetilde{s_j}$. Thus, $\widetilde{s_k}$ is the same as at most one of $\widetilde{s_i}$ and $\widetilde{s_j}$. Thus, $\delta(r_i, r_k) + \delta(r_j, r_k) \geq 1$. Thus, when $\delta(r_i, r_j) = 1$, it is also correct that $\delta(r_i, r_k) + \delta(r_j, r_k) \geq 1$. Thus, the theorem is proved. □

Fortunately, a consensus ranking is a full ranking. Then, we use the full-ranking triangle inequality to prove the lower bound of the optimal answer for the SWRA problem.

Theorem III.4. *There is a SWRA problem instance having n base rankings where γ base rankings conflicts with other base rankings, the maximum number of base rankings that a base ranking conflicts with is δ_{max}, and the average number of base rankings that a base ranking conflicts with is δ_{avg}. Suppose π^* is the optimal answer for the SWRA problem. Then, $D(\pi^*, R) \geq \frac{\gamma \cdot \delta_{avg}}{2 \cdot \delta_{max}}$.*

Proof. Suppose $R^{\#}$ is the set of base rankings conflicting with other rankings. Thus, $|R^{\#}| = \gamma$. By Theorem III.3, for any two rankings, r_i and r_j, in $R^{\#}$, we have $\delta(r_i, \pi^*) + \delta(r_j, \pi^*) \geq \delta(r_i, r_j) = 1$. Suppose $R_i^{\#}$ is the set of base rankings conflicting with r_i. Suppose D is the number of incompatible ranking pairs in R, i.e., $D = \frac{\gamma \cdot \delta_{avg}}{2}$. Besides, $\forall r_i \in R^{\#}$, $|R_i^{\#}| \leq \delta_{max}$ and $\sum_{r_i \in R^{\#}} |R_i^{\#}| = 2 \cdot D$. Therefore, $\sum_{r_i \in R^{\#}} \sum_{r_j \in R_i^{\#}} (\delta(r_i, \pi^*) + \delta(r_j, \pi^*)) \geq \sum_{r_i \in R^{\#}} \sum_{r_j \in R_i^{\#}} \delta(r_i, r_j) \geq 2 \cdot D$. Since $\forall r_i \in R^{\#}$, $|R_i^{\#}| \leq \delta_{max}$, we have $\sum_{r_i \in R^{\#}} \sum_{r_j \in R_i^{\#}} (\delta(r_i, \pi^*) + \delta(r_j, \pi^*)) \leq 2 \cdot \delta_{max} \cdot \sum_{r_i \in R^{\#}} \delta(r_i, \pi^*)$. Besides, $\forall r_i \in R \backslash R^{\#}$, $\delta(r_i, \pi^*) \geq 0$. Thus, $2 \cdot \delta_{max} \cdot D(\pi^*, R) = 2 \cdot \delta_{max} \cdot \left(\sum_{r_i \in R^{\#}} \delta(r_i, \pi^*) + \sum_{r_i \in R \backslash R^{\#}} \delta(r_i, \pi^*) \right) \geq \sum_{r_i \in R^{\#}} \sum_{r_j \in R_i^{\#}} (\delta(r_i, \pi^*) + \delta(r_j, \pi^*)) \geq 2 \cdot D$. Thus, $D(\pi^*, R) \geq \frac{D}{\delta_{max}} = \frac{\gamma \cdot \delta_{avg}}{2 \cdot \delta_{max}}$, which completes our proof. □

For instance, in Example 2, $R^{\#} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$. Thus, $\gamma = 7$. Besides, since r_5 is incompatible with r_1, r_2, r_3, r_6, and r_7, $\delta_{max} = 5$. In addition, since $D = 26$, $\delta_{avg} = \frac{26}{7}$. The minimum $D(\pi, R)$ of an answer for the SWRA problem instance in Example 2 is $4 \geq \frac{\gamma \cdot \delta_{avg}}{2 \cdot \delta_{max}} = 2.6$, which fits Theorem III.4. Next, based on Theorem III.4, we prove the approximation ratio of the CrossMerge algorithm.

Theorem III.5. *The approximation ratio of the CrossMerge algorithm is $2 \cdot \frac{\delta_{max}}{\delta_{avg}}$, where δ_{max} is the maximum number of base rankings that a base ranking conflicts with, and δ_{avg} is the average number of base rankings that a base ranking conflicts with.*

Proof. Suppose π_{cm} is the consensus ranking obtained by the CrossMerge algorithm, and π^* is the answer for the SWRA problem with a minimum $D(\pi^*, R)$. Suppose γ is the number of base rankings in R conflicting with other base rankings. Thus, $D(\pi_{cm}, R) \leq \gamma$. As proved in Theorem III.4, $D(\pi^*, R) \geq \frac{\gamma \cdot \delta_{avg}}{2 \cdot \delta_{max}}$. Thus, the approximation ratio is $\frac{D(\pi_{cm}, R)}{D(\pi^*, R)} \leq 2 \cdot \frac{\delta_{max}}{\delta_{avg}}$. □

For instance, in Example 2, $\frac{D(\pi_{cm}, R)}{D(\pi^*, R)} = 1 \leq 2 \cdot \frac{\delta_{max}}{\delta_{avg}}$, which fits Theorem III.5. The approximation ratio is related to the divergence between the base rankings. When the divergence increases, $\frac{\delta_{max}}{\delta_{avg}}$ is larger, and it is harder to get a near-optimal result, which is reasoning. After proving that the results of the CorssMerge algorithm are theoretically guaranteed, we next prove that the CrossMerge algorithm can terminate within polynomial time.

Theorem III.6. *The time complexity of the CrossMerge algorithm is $\mathcal{O}(n^3 \cdot m^2)$, where n is the number of base rankings in R, and m is the number of transactions in the transaction universe T.*

Proof. Since a base ranking may be a full ranking ordering m transactions, the time complexities of merging two rankings (i.e., line 4) and calculating the SWKTD between two rankings are both $\mathcal{O}(m^2)$. Since there are at most n base rankings in R, the time complexity of calculating $D(\pi_i, R)$ (i.e., line 5) is $\mathcal{O}(n \cdot m^2)$. Thus, the time complexity of the for-loop at line 3 is $\mathcal{O}(n^2 \cdot m^2)$. Since in each round of the while-loop (i.e., line 2), at least one base ranking will be removed from R', the while-loop runs at most n rounds. Thus, the time complexity of the while-loop is $\mathcal{O}(n^3 \cdot m^2)$. The time complexity of the last for-loop (i.e., line 10) is $\mathcal{O}(n)$. Thus, the time complexity of the CrossMerge algorithm is $\mathcal{O}(n^3 \cdot m^2)$. □

In other words, the time complexity of the CrossMerge algorithm is proportional to the cubic of the number of base rankings and the square of the number of transactions.

IV. The BubbleRank Algotihm

Although the CrossMerge algorithm can retrieve a near-optimal answer for the SWRA problem, its time complexity is high. Thus, we propose a more efficient approximation algorithm, the BubbleRank algorithm, which adds transactions one by one to the consensus ranking in a greedy manner. In this section, we will first introduce the basic idea of the

Algorithm 2: The BubbleRank algorithm.

Input: a transaction universe T and a set of base rankings R
Output: a consensus ranking π

```
1 π = ∅;
2 while T' ≠ ∅ do
3     foreach t_i ∈ T' do
4         π_i ← add t_i to the end of π;
5         clculate α_i = D(π_i, R);
6     π ← π_i with the lowest α_i, remove t_i from T';
7 return π;
```

BubbleRank algorithm. Then, we describe the BubbleRank algorithm in detail and demonstrate a running example. Finally, we theoretically analyze the time complexity as well as the approximation of the BubbleRank algorithm.

A. Basic Idea

The BubbleRank algorithm is inspired by the following two observations:

Observation 1: Except for the inherent time complexity of calculating $D(\pi, R)$, the time complexity of the CrossMerge algorithm is dominated by the number of times that a new consensus ranking is tried. The CrossMerge algorithm explores a new consensus ranking by merging base ranking in R, whose time complexity is related to the number of base rankings.

Observation 2: The number of base rankings in R can be much larger than the number of transactions in T. Given a transaction universe with m transactions, there are $\sum_{i=2}^{m} A_m^i$ different base rankings, where A_m^i is the number of permutations of m objects taken i at a time [26].

Thus, if we explore a new consensus ranking by finding the transaction with the next priority, the number of times that a new consensus ranking is tried is related to the number of transactions, which will be much smaller.

B. Algorithm Description

Based on the basic ideas in Subsection IV-A, we design the BubbleRank algorithm, which repeatedly adds a transaction to the end of the current consensus ranking in a greedy manner.

Algorithm 2 illustrates the pseudo-code of the BubbleRank algorithm. We first initialize the consensus ranking π as empty (line 1). Then, we add transactions into the consensus ranking π one by one in a greedy manner (line 2-6). Specifically, in the j^{th} round of the while-loop (line 2), we add a transaction to the consensus ranking π with the j^{th} priority. For each unordered transaction t_i in T, we first attempt to add it to the end of π, and the obtained updated π is denoted by π_i (line 4-5). We calculate the sum of SWKTDs, $D(\pi_i, R)$, between π_i and the base rankings in R, denoted by α_i. (line 5). Suppose π only contains t_j, r_i contains t_k and t_j, and t_k has a higher priority than t_j in r_i. Note that since the BubbleRank only adds a transaction to the end of the consensus ranking, although π has not contained t_k yet, the SWKTD between π and r_i is 1. After that, we update the consensus ranking as the ranking π_i with the lowest α_i and remove t_i from T' (line 6). Finally, we return π as the consensus ranking (line 7). Here is a running example of running the BubbleRank algorithm to retrieve the answer for the SWRA problem instance in Example 2.

Example 3 (Running Example)**.** *Initially, $\pi = \emptyset$, $T = \{t_1, t_2, t_3, t_4\}$. Then, in the first round of the while-loop, $\forall i \in [1, 4], \pi_i = \langle t_i \rangle$. Thus, $\alpha_1 = 1$, $\alpha_2 = 4$, $\alpha_3 = 3$, and $\alpha_4 = 4$. Thus, we update π as $\pi_1 = \langle t_1 \rangle$, and update T as $\{t_2, t_3, t_4\}$. Next, in the second round of the while-loop, $\pi_2 = \langle t_1 \succ t_2 \rangle$, $\pi_3 = \langle t_1 \succ t_3 \rangle$, and $\pi_4 = \langle t_1 \succ t_4 \rangle$. Thus, $\alpha_2 = 4$, $\alpha_3 = 4$, and $\alpha_4 = 4$. Thus, we update π as $\pi_2 = \langle t_1 \succ t_2 \rangle$, and update T as $\{t_3, t_4\}$. Next, in the third round of the while-loop, $\pi_3 = \langle t_1 \succ t_2 \succ t_3 \rangle$, and $\pi_4 = \langle t_1 \succ t_2 \succ t_4 \rangle$. Thus, $\alpha_3 = 5$ and $\alpha_4 = 5$. Thus, we update π as $\pi_3 = \langle t_1 \succ t_2 \succ t_3 \rangle$, and update T as $\{t_4\}$. In the fourth while-loop, $\pi_4 = \langle t_1 \succ t_2 \succ t_3 \succ t_4 \rangle$ and $\alpha_4 = 5$. Thus, we update π as $\pi_4 = \langle t_1 \succ t_2 \succ t_3 \succ t_4 \rangle$ and update T as empty. Thus, the while-loop terminates, and the consensus ranking returned by the BubbleRank algorithm is $\langle t_1 \succ t_2 \succ t_3 \succ t_4 \rangle$, whose $D(\pi, R)$ is 5.*

C. Theoretical Analysis

In this section, we theoretically analyze the time complexity as well as the approximation ratio of the BubbleRank algorithm.

Theorem IV.1. *The time complexity of the BubbleRank algorithm is $\mathcal{O}(n \cdot m^4)$, where n is the number of base rankings in R and m is the number of transactions in T.*

Proof. Initially, there are m transactions in T'. In each round of the while-loop, one transaction is removed from T'. Thus, the while-loop runs m rounds. Since there are n base rankings in R, and each base ranking has at most m transactions, the time complexity of calculating α_i is $\mathcal{O}(n \cdot m^2)$. Since there are at most n transactions in T', the total time complexity of the BubbleRank is $\mathcal{O}(n \cdot m^4)$. □

In other words, the time complexity of the BubbleRank algorithm increases linearly with the number of base rankings and the fourth power of the number of transactions. As we observed in Subsection IV-A, n can be much larger than m. Thus, the time complexity of the BubbleRank is smaller than that of the CrossMerge algorithm. For instance, for the SWRA problem instance in Example 2, there are seven base rankings and four transactions. The CrossMerge algorithm calculates d_i 13 times (as demonstrated in Example 2), but the BubbleRank algorithm calculates d_i only 10 times (as demonstrated in Example 3). After proving the time complexity of the BubbleRank algorithm, we next prove its approximation ratio.

Theorem IV.2. *The approximation ratio of the BubbleRank algorithm is $\mathcal{O}(\frac{l_{max}}{l_{min}} \cdot \frac{\delta_{max}}{\delta_{min}})$, where $l_{max/min}$ is the maximum/minimum length of a base ranking, and $\delta_{max/min}$ is the maximum/minimum number of base rankings that a base ranking conflicting with.*

Proof. Suppose $R^{\#}$ is the set of base rankings that having incompatible base rankings in R. Suppose $T^{\#}$ is the union of the transactions contained in base rankings in $R^{\#}$, i.e., $T^{\#} = \{t_i | t_i \in r_i, r_i \in R^{\#}\}$. Suppose π^* is the consensus ranking with a minimum $D(\pi^*, R)$. Thus, π^* contains at least $\frac{|T^{\#}|}{l_{max}}$ base rankings in $R^{\#}$. Thus, $D(\pi^*, R) \geq \frac{|T^{\#}|}{l_{max}} \cdot \delta_{min}$. Suppose π_{br} is the consensus ranking returned by the BubbleRank algorithm. Since π_{br} contains at most $\frac{|T^{\#}|}{l_{min}}$ base rankings in $R^{\#}$, $D(\pi_{br}, R) \leq \frac{|T^{\#}|}{l_{min}} \cdot \delta_{max}$. Thus, the approximation ratio is $\frac{D(\pi_{br},R)}{D(\pi^*,R)} \leq \frac{l_{max}}{l_{min}} \cdot \frac{\delta_{max}}{\delta_{min}}$, which completes the proof. □

For instance, in Example 2, the maximum length of a base ranking is $l_{max} = |r_1| = 3$, and the minimum length of a base ranking is $l_4 = |r_4| = 2$. Besides, the maximum number of base rankings that a base ranking conflicting with is $\delta_{max} = 5$, i.e., r_5 conflicts with r_1, r_2, r_3, r_6, and r_7. Besides, the minimum number of base rankings that a base ranking conflicting with is $\delta_{min} = 2$, i.e., r_6 only conflicts with r_5 and r_7. The minimum $D(\pi, R)$ of an answer for the SWRA problem instance in Example 2 is 4. As demonstrated in Example 3, $D(\pi_{br}, R) = 5$. Thus, $\frac{D(\pi_{br},R)}{D(\pi^*,R)} = 1.25 \leq \frac{l_{max}}{l_{min}} \cdot \frac{\delta_{max}}{\delta_{min}} = 3.75$, which fits Theorem IV.2.

REFERENCES

[1] J. d. Borda, "Mémoire sur les élections au scrutin," *Histoire de l'Academie Royale des Sciences pour 1781 (Paris, 1784)*, 1784.

[2] N. De Condorcet, *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge University Press, 2014.

[3] T. Jin, P. Xu, Q. Gu, and F. Farnoud, "Rank aggregation via heterogeneous thurstone preference models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 4353–4360, 2020.

[4] Q. Xu, Z. Yang, Z. Chen, Y. Jiang, X. Cao, Y. Yao, and Q. Huang, "Deep partial rank aggregation for personalized attributes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 678–688, 2021.

[5] C. Kuhlman and E. Rundensteiner, "Rank aggregation algorithms for fair consensus," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2706–2719, 2020.

[6] F. M. Choudhury, Z. Bao, J. S. Culpepper, and T. Sellis, "Monitoring the top-m rank aggregation of spatial objects in streaming queries," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 585–596, IEEE, 2017.

[7] C. Domshlak, A. Gal, and H. Roitman, "Rank aggregation for automatic schema matching," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 4, pp. 538–553, 2007.

[8] M. A. Soliman and I. F. Ilyas, "Ranking with uncertain scores," in *2009 IEEE 25th international conference on data engineering*, pp. 317–328, IEEE, 2009.

[9] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[10] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," *arXiv preprint arXiv:1904.05234*, 2019.

[11] P. Ruan, T. T. A. Dinh, Q. Lin, M. Zhang, G. Chen, and B. C. Ooi, "Lineagechain: a fine-grained, secure and efficient data provenance system for blockchains," *The VLDB Journal*, vol. 30, no. 1, pp. 3–24, 2021.

[12] P. Ruan, D. Loghin, Q.-T. Ta, M. Zhang, G. Chen, and B. C. Ooi, "A transactional perspective on execute-order-validate blockchains," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 543–557, 2020.

[13] Y. Zhang, S. Setty, Q. Chen, L. Zhou, and L. Alvisi, "Byzantine ordered consensus without byzantine oligarchy," in *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, pp. 633–649, 2020.

[14] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the 10th international conference on World Wide Web*, pp. 613–622, 2001.

[15] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, "Comparing partial rankings," *SIAM Journal on Discrete Mathematics*, vol. 20, no. 3, pp. 628–648, 2006.

[16] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, "Comparing and aggregating rankings with ties," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 47–58, 2004.

[17] H. Gilbert, T. Portoleau, and O. Spanjaard, "Beyond pairwise comparisons in social choice: A setwise kemeny aggregation problem," *Theoretical Computer Science*, 2021.

[18] D. W. Jacobs, D. Weinshall, and Y. Gdalyahu, "Classification with nonmetric distances: Image retrieval and class representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 583–600, 2000.

[19] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th international conference on data engineering*, pp. 673–684, IEEE, 2002.

[20] S.-W. Kim, S. Park, and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," in *Proceedings 17th International Conference on Data Engineering*, pp. 607–614, IEEE, 2001.

[21] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 491–502, 2005.

[22] H. P. Young and A. Levenglick, "A consistent extension of condorcet's election principle," *SIAM Journal on applied Mathematics*, vol. 35, no. 2, pp. 285–300, 1978.

[23] J. Bartholdi, C. A. Tovey, and M. A. Trick, "Voting schemes for which it can be difficult to tell who won the election," *Social Choice and welfare*, vol. 6, no. 2, pp. 157–165, 1989.

[24] L. Chen and X. Lian, "Efficient similarity search in nonmetric spaces with local constant embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 3, pp. 321–336, 2008.

[25] X. Lian and L. Chen, "Trip planner over probabilistic time-dependent road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 2058–2071, 2013.

[26] S. M. Ross, *Introduction to probability and statistics for engineers and scientists*. Academic Press, 2020.