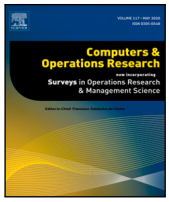
Computers & Operations Research 153 (2023) 106164

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



Approximate Condorcet Partitioning: Solving large-scale rank aggregation problems

Sina Akbari *, Adolfo R. Escobedo

School of Computing and Augmented Intelligence, Arizona State University, P.O. Box 878809, Tempe, AZ 85287-8809, United States

ARTICLE INFO

Keywords:
Group decision-making
Rank aggregation
Computational social choice
Condorcet criterion
Kemeny–Snell distance

ABSTRACT

Rank aggregation has ubiquitous applications in computer science, operations research, and various other fields. Most attention on this problem has focused on an NP-hard variant known as Kemeny aggregation, for which solution approaches with provable guarantees that can handle difficult high-dimensional instances remain elusive. This work introduces exact and approximate methodologies inspired by the social choice foundations of the problem, namely the Condorcet Criterion. We formalize the concept of the finest-Condorcet partition for rankings that may contain ties and specify its required conditions. We prove that this partition is unique and devise an efficient algorithm to obtain it. To deal with instances where it does not yield many subsets, we propose Approximate Condorcet Partitioning (ACP), with which larger subsets can be further broken down and more easily solved. ACP is a scalable solution technique capable of handling large instances while still providing provable guarantees. Although ACP approximation factors are instance-specific, their values were lower than those offered by all known constant-factor approximation schemes — inexact algorithms whose resulting objective values are guaranteed to be within a specified fixed percent of the optimal objective value — for all 113 instances tested herein (containing up to 2,820 items). What is more, ACP obtained solutions that deviated by at most two percent from the optimal objective function values for a large majority of these instances.

1. Introduction

Rank aggregation is a well studied problem in operations research, computer science, and computational social choice, which arises in a variety of situations where m judges are asked to rank n items based on some quality of interest. The objective is to find an *aggregate ranking*, defined as the ranking of items with the lowest cumulative disagreement to the given rankings — prior related works often refer to this as the *consensus ranking*, but we eschew use of this term to avoid the implication that the solution ranking does not cause disagreement. Rank aggregation has been utilized in wide-ranging real-world situations. A recent application is the pool riding problem, whose goal is to assign a pool of passengers to a set of drivers with predetermined routes. Drivers are ranked for each passenger based on different criteria such as percentage of matched routes and initial distance; the final ranking is obtained by consolidating the rankings associated with each criterion. Rank aggregation has been shown to increase the passenger–driver matching rate (Şahin et al., 2022). Another popular application is within the field of bioinformatics, specifically, to accurately identify and rank genes possibly related to a disease from micro-array gene expression data (Mandal and Mukhopadhyay, 2017; Wald et al., 2012). Since various statistical tools and data mining techniques can be applied for this purpose, rank aggregation has been utilized to avoid relying on any single technique or study. On a similar direction, rank aggregation is being increasingly utilized as an ensemble technique to consolidate output rankings of different machine learning algorithms seeking to evaluate a common set of entities. One such example is the label ranking problem, whose objective is to predict the ranking of a set of labels, given a set of input attributes. In this context, the output of different label ranking algorithms can be aggregated to produce a more robust prediction (Aledo et al., 2017a). Similar uses in related contexts include feature selection (Dahiya et al., 2016; Drotár et al., 2019), natural language processing (Cascaro et al., 2019; Onan, 2018), recommender systems (Oliveira et al., 2020), and data query (Cohen-Boulakia et al., 2011). There are also wider applications in meta-search engines (Desarkar et al., 2016; Dwork et al., 2001), crowdsourcing and human computation (Kemmer et al., 2020; Mao et al., 2013), multi-criteria decision-making (Dong et al., 2021; Mohammadi and Rezaei, 2020), and network inference (Marbach et al., 2012; Puerta et al., 2021).

* Corresponding author.
E-mail addresses: sina.akbari@asu.edu (S. Akbari), adres@asu.edu (A.R. Escobedo).

https://doi.org/10.1016/j.cor.2023.106164
Received 25 May 2022; Received in revised form 21 November 2022; Accepted 26 January 2023
Available online 2 February 2023

One of the principled ways to address the rank aggregation problem is by using distances founded on rigorous mathematical axioms (Brandt et al., 2016; Cook, 2006). Kemeny and Snell (1962) introduced the first such framework. The authors proposed a set of axioms — non-negativity, triangular inequality, anonymity, extension, scaling, and commutativity — that should be satisfied by any distance metric on rankings, and they introduced a distance that uniquely satisfies them. The optimization problem induced with this distance metric is known as the Kemeny aggregation problem (Kemeny-Agg). The aggregate ranking returned by Kemeny-Agg satisfies various key social choice properties such as neutrality, local stability, and the Condorcet Criterion (Brandt et al., 2016; Young and Levenglick, 1978), which translate into practical benefits such as spam detection (Dwork et al., 2001). These theoretical benefits come at a high computational price as Kemeny-Agg is NP-hard when there are four or more input rankings (Bartholdi et al., 1989; Dwork et al., 2001). It is worth adding that non-strict rankings (i.e., rankings that *may* contain ties) are more numerous than strict rankings (i.e., rankings without ties) over the same set of items — $n!$ strict rankings compared to approximately $0.5n!(1.4)^{n+1}$ non-strict rankings (Gross, 1962). Hence, their aggregation is much more difficult, i.e., due to a higher number of possible solution rankings that must be considered.

Kemeny-Agg has been modeled and solved using binary programming formulation in Conitzer et al. (2006), Cook (2006) and Yoo and Escobedo (2021). Other exact methods include the specialized branch and bound algorithm of Emond and Mason (2002) and the exact implicit enumeration algorithm of Azzini and Munda (2020). Exact methods can solve only small- to medium-sized instances; the largest instances solved exactly, for example, in Azzini and Munda (2020), Emond and Mason (2002), Conitzer et al. (2006), Betzler et al. (2014), and Yoo and Escobedo (2021) had 14, 15, 40, 200, and 210 items, respectively. Due to the general computational intractability of Kemeny-Agg, large-scale instances have been solved nearly exclusively via inexact methods. These include various heuristics such as those of Aledo et al. (2017b, 2019), Badal and Das (2018), D'Ambrosio et al. (2017) and Ding et al. (2018), to name but a few. Although many of them are designed to compute promising solutions quickly, heuristics generally do not provide formal guarantees on the solution quality. Hence, various researchers have also focused on developing approximation schemes – algorithms whose resulting objective values are guaranteed to be within a specified fixed percent of the objective value – which will serve as the basis of comparison for the methods developed herein. The two ensuing paragraphs provide a review of notable approximation algorithms for solving Kemeny-Agg.

First, we review approximation algorithms designed for Kemeny-Agg with strict rankings. The aggregate list according to Spearman's footrule distance is a 2-approximation algorithm (Dwork et al., 2001) for Kemeny-Agg; this distance calculates the sum of the absolute differences between the rank positions assigned to each of the items. Ailon et al. (2008) proposed *KwikSort* and *LP-KwikSort*. KwikSort, an expected 2-approximation algorithm, repeatedly chooses a random item as the pivot, and it divides the remaining items into two groups — the sets of items ranked ahead and behind the pivot item — based on the pairwise comparison information; LP-KwikSort also chooses a random item as the pivot, and it divides the remaining items into two groups based on the linear programming (LP) relaxation solution of a Kemeny-Agg formulation. The authors proved that the best of KwikSort and *Pick-A-Perm* (see the next paragraph) yields an expected 11/7-approximation, and the best of LP-KwikSort and Pick-A-Perm yields an expected 4/3-approximation. In effect, KwikSort and LP-KwikSort do well on instances in which the Pick-A-Perm does not, and vice versa. Kenyon-Mathieu and Schudy (2007) derived the first polynomial time approximation scheme (PTAS) for the feedback arc set problem (FASP) on tournaments; a PTAS is a $(1+\epsilon)$-approximation — i.e., it returns a solution up to $(1+\epsilon)$ times the optimal objective function value, for any fixed value of $\epsilon > 0$. The authors also introduced a weighted generalization of the PTAS for FASP to provide the first PTAS for Kemeny-Agg. Because the time complexity of this PTAS is doubly exponential in $1/\epsilon$, its implementation becomes impractical for sufficiently small ϵ (Betzler et al., 2014).

Second, we review approximation algorithms suitable for strict and non-strict rankings. Pick-A-Perm (Ailon et al., 2008) selects one of the input rankings at random, and its deterministic version called *BestInput*, picks the input ranking with the lowest cumulative Kemeny–Snell distance to the input rankings. Pick-A-Perm and BestInput are expected 2-approximation algorithms for strict rankings; however, their approximation factors have not yet been defined for the case of non-strict rankings. Ailon (2010) proposed *RepeatChoice*, an expected 2-approximation algorithm, and *LPKwikSort*$_h$, an expected 3/2-approximation. While these two algorithms allow the input rankings to be non-strict, the aggregate ranking is required to be strict, which is not be suitable for many applications. RepeatChoice repeatedly and without replacement chooses an input ranking and refines an initial non-strict ranking until all ties are broken. LPKwikSort$_h$ uses a novel LP rounding technique and is the first algorithm that, by itself, provides an (expected) approximation factor lower than 2. Van Zuylen and Williamson (2007) proposed a derandomized version of KwikSort, referred to herein as *DeterministicKwikSort*, which is an expected 2-approximation algorithm, and showed that the best solution achieved by their algorithm and RepeatChoice provides an expected 8/5-approximation. Their work, similar to Ailon (2010), allows the input rankings to be non-strict but not the aggregate ranking.

Next, we turn to partitioning algorithms that have been developed to expedite exact methods for Kemeny-Agg based on key social choice properties guaranteed to be satisfied by the optimal solution(s) to this problem. A small number of works have introduced theory and algorithms that enable certain large instances to be decomposed into a set of smaller subproblems while guaranteeing that solving them independently still induces an optimal solution to the original problem. More specifically, these specialized partitioning methods efficiently determine the relative ordering of subsets of items; however, the exact ordering of the items within each individual subset is determined by solving a Kemeny-Agg subproblem restricted to that subset. The final ranking is obtained by properly concatenating the shorter rankings of items within each subset according to how the subsets were ordered in the partitioning. These partitioning schemes have been utilized to accelerate exact formulations (Betzler et al., 2014; Schalekamp and Zuylen, 2009; Yoo and Escobedo, 2021) and lower bounding techniques (Akbari and Escobedo, 2021) of Kemeny-Agg. A notable scheme is based on the Extended Condorcet Criterion (XCC) proposed by Truchon (1998), who proved that the optimal solutions to Kemeny-Agg for strict rankings are consistent with XCC and devised a partitioning algorithm that takes advantage of this theoretical guarantee. Betzler et al. (2014) introduced another scheme based on the *3/4-Majority Rule*. The authors proved that this different type of partitioning scheme cannot further partition an instance that has already been partitioned by XCC, meaning that XCC partitioning is always at least as good as partitioning using the 3/4-Majority Rule. Yoo and Escobedo (2021) showed that XCC is inconsistent with the solution to Kemeny-Agg with non-strict rankings, that is, the optimal solution to Kemeny-Agg may violate XCC. In the light of this fact, they proposed the Non-strict Extended Condorcet Criterion (NXCC), a generalization of XCC suitable for both strict and non-strict rankings, and introduced a partitioning algorithm based on sequential pairwise comparisons. The authors reported that whenever instances from the Preflib data set (Mattei and Walsh, 2013) with up to 210 items were partitionable, the combined exact solution times of the decomposed subproblems — using their exact binary programming formulation — were at least 25% and up to 96% faster than those of the full problem. Betzler et al. (2014) reported similar results on the effectiveness of XCC with up to a 35x computational speedup. Recently, Akbari and Escobedo (2021) reported that NXCC rendered up to a 25x computational improvement when implemented to compute

lower bounds for Kemeny-Agg with non-strict rankings. It is worth mentioning that a related yet significantly distinct approach introduced by Milosz and Hamel (2018, 2020) aims to find the relative ordering of certain item-pairs in the aggregate ranking. While this approach was shown to be more effective than XCC in practice, it is only applicable for strict rankings and it has a time complexity of $O(n^3)$, whereas XCC has a time complexity of $O(n^2)$.

This work makes three main contributions. First, it improves XCC- and NXCC-based decomposition by defining the finest possible partition that is consistent with these properties. This *finest-Condorcet partition* yields the most subsets among all such possible partitions, thereby maximizing their potential computational benefits. Second, it derives an efficient algorithm for obtaining the finest-Condorcet partition. Even though, XCC- and NXCC-based decomposition are useful on certain instances, not all instances are partitionable (Betzler et al., 2014; Yoo and Escobedo, 2021) and when they are, there may be one or a few large subsets that are difficult to solve to optimality, hence limiting the usefulness of these schemes in practice. Motivated by this fact, the present work introduces as its third main contribution *Approximate Condorcet Partitioning (ACP)*, a scalable solution approach for Kemeny-Agg with provable guarantees suitable for high-dimensional instances. ACP attempts to further decompose the finest-Condorcet partition whenever there are one or more subsets (i.e., subproblems) that are too large to solve using exact methods. This contribution is accompanied by the derivation of instance-specific approximation factors, which are applicable to any item-partitioning scheme, including those that may not be consistent with Condorcet extensions. Improved guarantees are derived for the proposed solution technique via ACP. Although these approximation factors are also instance-specific, their values were lower than those offered by all constant-factor approximation algorithms known to date, for all tested instances herein.

The rest of this paper is organized as follows. Section 2 introduces the notations used throughout this paper, provides some preliminary definitions, and reviews the Condorcet Criterion and its extensions. Section 3 introduces the finest-Condorcet partition and proves its uniqueness; furthermore, it develops an efficient algorithm to construct it. Section 4 introduces the Approximate Condorcet Partitioning technique and derives its provable guarantees. Section 5 presents the results of comparing this technique with various prominent approximation algorithms. Finally, Section 6 concludes the paper.

2. Notation and preliminaries

Rankings can be categorized as strict and non-strict. Strict rankings refer to the case where there are no ties, while non-strict rankings refer to the case where there may be ties. It is worth emphasizing that set of non-strict rankings includes all strict rankings; stated otherwise, non-strict rankings do not necessarily include ties, rather they *may* contain ties. Both strict and non-strict rankings can further be categorized as complete and incomplete; all items are ranked in the former and some items may be unranked in the latter. This work focuses on complete non-strict rankings. However, all techniques proposed herein are applicable to strict rankings as well due to aforementioned relationship between strict and non-strict rankings. This section is organized as follows. Section 2.1 describes basic mathematical notations to introduce the rank aggregation problem, and Section 2.2 describes the social choice-inspired decomposition schemes that serve as the foundations of this paper.

2.1. Mathematical notation

Let $\mathcal{X} = \{1, 2, \dots, n\}$ be the set of items, $\mathcal{L} = \{1, 2, \dots, m\}$ be the set of indices of input rankings over $\mathcal{X}$, and σ^l be the lth input ranking, where $l \in \mathcal{L}$. Additionally, let $\Sigma \subset \mathbb{Z}^n$ be the set of all possible complete ranking vectors over $\mathcal{X}$, and σ_i^l be the rank of item i in σ^l. As a convention, $i \succ_\sigma j$ indicates that item i is preferred over item j in σ (i.e., $\sigma_i < \sigma_j$), and $i \approx_\sigma j$ indicates that i and j are tied in σ (i.e., $\sigma_i = \sigma_j$). Additionally, let a *full rank reversal* denote the case where two rankings σ^1, σ^2 fully disagree over the relative orderings of i and j (one of them ranks i ahead of j, and the other ranks j ahead of i); additionally, let a *partial rank reversal* denote the case where i and j are tied in one ranking, but not in the other.

Definition 1. The Kemeny–Snell distance between two complete rankings σ^1, σ^2, denoted by $d_{KS}(\sigma^1, \sigma^2)$, is given by

$$d_{KS}(\sigma^1, \sigma^2) = \frac{1}{2} \sum_{i \in \mathcal{X}} \sum_{j \in \mathcal{X}} \left| \text{sign}(\sigma_i^1 - \sigma_j^1) - \text{sign}(\sigma_i^2 - \sigma_j^2) \right|. \tag{1}$$

The function $sign(v)$ returns 1 if $v > 0$, -1 if $v < 0$, and 0 otherwise. In the case of strict rankings, distance d_{KS} counts the number of full rank reversals; in the case of non-strict rankings, it assigns a weight of two for each full rank reversal and a weight of one for every partial rank reversal.

Definition 2. The aggregate ranking obtained from Kemeny-Agg can be mathematically stated as:

$$\sigma^* = \underset{\sigma \in \Sigma}{\text{argmin}} \sum_{l \in \mathcal{L}} d_{KS}(\sigma, \sigma^l). \tag{2}$$

Definition 3. Let $s_{ij} = |l \in \mathcal{L} : i \succ_{\sigma^l} j|$ and $t_{ij} = |l \in \mathcal{L} : i \approx_{\sigma^l} j|$ be the number of input rankings in which item i is preferred over item j, and the number of input rankings in which i and j are tied, respectively.

Definition 4 (*Yoo and Escobedo, 2021*)**.** Item i is pairwise preferred by a decisive majority over item j if $s_{ij} > s_{ji} + t_{ij}$, that is, the number of input rankings which prefer i to j is greater than the number of input rankings which prefer j to i, plus those which tie them. If neither i is preferred over j nor j is preferred over i, then there is no decisive majority that prefers i over j or j over i.

For succinctness, the rest of the paper employs the term *pairwise preferred* as shorthand for *pairwise preferred by a decisive majority*.

Definition 5. Let $d_{KS}(\sigma)$ be the cumulative Kemeny–Snell distance of a given ranking $\sigma \in \Sigma$ to the input rankings; it is useful to also expand $d_{KS}(\sigma)$ as $\sum_{i \in X} \sum_{j \in X} d_{KS}(\sigma_{ij})$, where $d_{KS}(\sigma_{ij})$ is the contribution of each pair of items (i, j) in $d_{KS}(\sigma)$, which is given by

$$d_{KS}(\sigma_{ij}) = \begin{cases} 2s_{ji} + t_{ij} & \text{if} \quad i \succ_\sigma j, \\ 2s_{ij} + t_{ij} & \text{if} \quad j \succ_\sigma i, \\ s_{ij} + s_{ji} & \text{if} \quad i \approx_\sigma j. \end{cases} \tag{3}$$

Eq. (3) follows from the definition of the Kemeny–Snell distance function. As stated earlier, distance d_{KS} assigns a weight of 2 for each rank reversal and a weight of 1 for every partial rank reversal. Therefore, if i is ranked ahead of j in σ, the imposed distance for this pair equals the number of input rankings where j is ranked ahead of i, times 2, plus the number of input rankings where i and j are tied. Furthermore, if i and j are tied in σ, the imposed distance for this pair equals the number of input rankings where either i is ranked ahead of j or vice versa.

Definition 6. Let $[c_{ij}] \in \mathbb{Z}^{n \times n}$ be the Cumulative Ranking (CR) matrix whose individual entries are obtained as $c_{ij} = s_{ij} + t_{ij} - s_{ji}$, when the input rankings are complete.

The CR matrix is used to linearize Kemeny-Agg formulation in Yoo and Escobedo (2021). Here, it is employed to reduce the space requirements of the proposed algorithm.

An ordered set of subsets $\mathbf{X} = \{X_1, X_2, \dots, X_w\}$ is a partition of $\mathcal{X}$ if $\cup_{k=1}^{w} X_k = \mathcal{X}$ and $X_k \cap X_{k'} = \emptyset, \forall k, k' \in \{1, \dots, w\}$, with $k \neq k'$. Subset X_k is said to be preferred over subset $X_{k'}$, written as $X_k \succ X_{k'}$, if all items in X_k are pairwise preferred over all items in $X_{k'}$. Similar to Laslier

(1997), we call partition $\mathbf{X}$ a *null* partition if $|\mathbf{X}| = 1$, a *trivial* partition if $|\mathbf{X}| = n$, and a *proper* partition otherwise.

We close this section with an example to illustrate the featured notation.

Example 1. Consider an instance with 6 rankings of 6 items. The input rankings and the pairwise comparison matrices, $\mathbf{S} = [s_{ij}] \in \mathbb{Z}^{6\times6}$, $\mathbf{T} = [t_{ij}] \in \mathbb{Z}^{6\times6}$, and $CR = [c_{ij}] \in \mathbb{Z}^{6\times6}$, are given by

Item	Input rankings					
	σ^1	σ^2	σ^3	σ^4	σ^5	σ^6
1	1	1	1	1	1	1
2	3	3	3	2	2	1
3	2	2	4	4	2	2
4	3	3	3	3	3	5
5	4	4	2	4	5	3
6	5	5	5	5	4	4

$$\mathbf{S} = \begin{bmatrix} 0 & 5 & 6 & 6 & 6 & 6 \\ 0 & 0 & 3 & 3 & 5 & 6 \\ 0 & 2 & 0 & 4 & 4 & 6 \\ 0 & 0 & 2 & 0 & 4 & 5 \\ 0 & 1 & 1 & 2 & 0 & 5 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{CR} = \begin{bmatrix} 0 & 6 & 6 & 6 & 6 & 6 \\ -4 & 0 & 2 & 6 & 4 & 6 \\ -6 & 0 & 0 & 2 & 4 & 6 \\ -6 & 0 & -2 & 0 & 2 & 4 \\ -6 & -4 & -2 & -2 & 0 & 4 \\ -6 & -6 & -6 & -4 & -4 & 0 \end{bmatrix}.$$

Consider σ^1 and σ^6. The two rankings fully disagree over the relative ordering of item-pairs (2, 3), (4, 5), and (4, 6), and they partially disagree over the relative ordering of item-pairs (1, 2) and (2, 4). Since d_{KS} assigns a weight of two to every full rank reversal and a weight of one to every partial rank reversal, we have $d_{KS}(\sigma^1, \sigma^6) = 8$. Furthermore, the cumulative distance between σ^1 and the rest of input rankings is given by

$$d_{KS}(\sigma^1) = \sum_{l=2}^{6} d_{KS}(\sigma^1, \sigma^l) = 0 + 10 + 6 + 4 + 8 = 28.$$

2.2. The Condorcet Criterion and its variants

The Condorcet Criterion (CC), first proposed by Marquis de Condorcet (1785), is among the most prominent social choice properties arising from voting theory. CC states that a candidate who is pairwise preferred over all other candidates must be declared as the top-ranked candidate, formally known as the *Condorcet Winner*. CC can be formally stated as (Young, 1988)

$$\text{if } \exists\, i \in X : \; s_{ij} > s_{ji} \quad \forall j \in \mathcal{X}\backslash\{i\} \implies i \succ_{\sigma} j \quad \forall j \in \mathcal{X}\backslash\{i\},$$

where σ is the aggregate ranking. A voting rule is said to be *Condorcet consistent* if it always selects the Condorcet Winner as the top-ranked item, when one exists (Brandt et al., 2016). There are other Condorcet consistent rank aggregation methods such as Dodgson's rule (Dodgson, 1876), maximin rule (Young, 1977), and ranked pairs rule (Tideman, 2017).

Truchon (1998) proposed the Extended Condorcet Criterion (XCC), which generalizes CC to guarantee an ordering of item-subsets in the aggregate ranking. XCC states that if $\mathcal{X}$ can be arranged into a partition $\mathbf{X} = \{X_1, X_2, \ldots, X_w\}$ such that $X_k \succ X_{k'}$, $\forall k, k' \in \{1, \ldots, w\}$, with $k < k'$, then all items in X_k must be ranked ahead of all items in $X_{k'}$ in the aggregate ranking. XCC can be stated formally as:

$$\text{if } \; s_{ij} > s_{ji} \quad \forall i \in X_k \;\; \forall j \in X_{k'} \;\; \forall k < k'$$
$$\implies \; i \succ_{\sigma^*} j \quad \forall i \in X_k \;\; \forall j \in X_{k'} \;\; \forall k < k',$$

where σ^* is the optimal solution to Kemeny-Agg. This means that, in the optimal solution, items belonging to lower-indexed subsets in the partition must be strictly ranked ahead of items belonging to higher-indexed subsets. This partitioning scheme can be very useful in practice since through these partitioning approaches, certain large instances can be decomposed into a set of smaller subproblems while guaranteeing that solving them independently still induces an optimal solution to the original problem. Note that the precise ordering of the items within each subset is not obtained by applying this property alone; it is determined by solving rank aggregation subproblems restricted to the items in each subset of the partition — this applies to all partitioning schemes discussed in this paper.

Recently, Yoo and Escobedo (2021) showed that Kemeny-Agg with non-strict rankings is inconsistent with XCC. That is, solutions to this problem, which allows rankings with and without ties, may violate XCC. The authors defined a social choice property called the Non-strict Extended Condorcet Criterion (NXCC), which can be stated formally as:

$$\text{if } \; s_{ij} > s_{ji} + t_{ij} \quad \forall i \in X_k \;\; \forall j \in X_{k'} \;\; \forall k < k'$$
$$\implies \; i \succ_{\sigma^*} j \quad \forall i \in X_k \;\; \forall j \in X_{k'} \;\; \forall k < k'.$$

Observe that, for the case with all strict rankings (i.e., $t_{ij} = 0 \; \forall i, j \in X$), NXCC becomes XCC. It was formally demonstrated in Yoo and Escobedo (2021) that the aggregate rankings returned by Kemeny-Agg for non-strict rankings are consistent with NXCC.

3. The finest-Condorcet partition

Henceforth, we will denote partitions based on XCC and NXCC simply as *Condorcet partitions*, to distinguish them from alternative partitioning schemes (e.g. see Betzler et al., 2014). The rest of this section is organized as follows. Section 3.1 formally introduces the concept of the finest-Condorcet partition, specifies its conditions, and proves that it is unique. Section 3.2 proposes a novel algorithm for obtaining the finest-Condorcet partition.

3.1. Definition and properties

Let $\wp(\mathcal{X})$ denote the class of partitions satisfying NXCC. This class can contain more than one partition; however, certain members of $\wp(\mathcal{X})$ are more computationally expedient than others. In particular, after obtaining a Condorcet partition, it is necessary to solve a Kemeny-Agg subproblem for each subset of the partition and then to concatenate the separate solutions, in proper order, to obtain a solution to the original problem. The worst case happens when the instance has a null Condorcet Partition ($|\mathbf{X}| = 1$), and the best case happens when the instance has a trivial Condorcet partition ($|\mathbf{X}| = n$) (i.e., the order of the singleton subsets in the partition provides the optimal ranking of all items). For this reason, it is desirable to obtain partitions with more subsets and/or with smaller subsets. The ensuing example illustrates the differences between multiple NXCC partitions and motivates our focus on the finest among all such partitions.

Example 2. Consider the instance given in Example 1. There are seven NXCC partitions: $\mathbf{X}^1 = \{\{1\}, \{2, 3, 4\}, \{5\}, \{6\}\}$, $\mathbf{X}^2 = \{\{1, 2, 3, 4\}, \{5\}, \{6\}\}$, $\mathbf{X}^3 = \{\{1, 2, 3, 4, 5\}, \{6\}\}$, $\mathbf{X}^4 = \{\{1\}, \{2, 3, 4\}, \{5, 6\}\}$, $\mathbf{X}^5 = \{\{1, 2, 3, 4\}, \{5, 6\}\}$, $\mathbf{X}^6 = \{\{1\}, \{2, 3, 4, 5\}, \{6\}\}$, and $\mathbf{X}^7 = \{\{1, 2, 3, 4, 5, 6\}\}$. The first is the finest and most desirable, as it has the highest number of subsets; indeed, the only subproblem that needs to be solved is the one corresponding to items 2, 3, 4 (since the other subsets are singletons). Notice that it is possible to further partition at least one subset in $\mathbf{X}^2$–$\mathbf{X}^7$ while satisfying NXCC. Partition $\mathbf{X}^1$ implies that item 1 will be ranked first, items 2, 3, and 4 will be ranked ahead of items 5 and 6, and item 5 will be ranked ahead of item 6 in the optimal solution to Kemeny-Agg; however, this partition on its own cannot determine the exact ordering of the three items in the second subset.

The concept of a *finest-Condorcet partition* was first introduced in Truchon (1998), although its formal definition or required conditions were not provided therein. This partition is an adaption of the unique *minimal decomposition* (Laslier, 1997) of a tournament, which is applicable only for aggregating strict rankings. Next, we formally define an extension of the finest-Condorcet partition that is suitable for both strict and non-strict rankings, and we specify its required conditions.

Definition 7. Partition $\mathbf{X}^f \in \wp(\mathcal{X})$ is the finest-Condorcet partition if there is no other partition $\mathbf{X} \in \wp(\mathcal{X})$ such that $|\mathbf{X}| > |\mathbf{X}^f|$, that is, $\mathbf{X}^f$ is the partition with the most subsets.

For any $\mathbf{X} \in \wp(\mathcal{X})\backslash\mathbf{X}^f$ — i.e., all but the finest partition of the class — it is possible to further decompose at least one of the subsets such that the resulting partition still satisfies NXCC. To improve both XCC and NXCC, we add a requirement that is only satisfied by $\mathbf{X}^f$. Let $\wp^f(\mathcal{X})$ be the class of finest-Condorcet partitions. Any $\mathbf{X}^f \in \wp^f(\mathcal{X})$ must satisfy

$$\forall X_k \in \mathbf{X}^f,\ \nexists \overline{X}_k \subset X_k :\ s_{ij} > s_{ji} + t_{ij}, \forall i \in \overline{X}_k, \forall j \in X_k\backslash\overline{X}_k. \tag{4}$$

Condition (4) does not allow a subset of items in X_k to be pairwise preferred over the rest of the items in X_k, for all $X_k \in \mathbf{X}$ (i.e., the subsets cannot be further decomposed while satisfying NXCC). Later we prove that NXCC and Condition (4) are the necessary and sufficient conditions for the finest-Condorcet partition. Beforehand, Theorem 1 proves that $|\wp^f(\mathcal{X})| = 1$, meaning that $\mathbf{X}^f$ is unique.

Theorem 1. *The finest-Condorcet partition is unique.*

Theorem 2. *$\mathbf{X}^f \in \wp(\mathcal{X})$ is the finest-Condorcet partition if and only if it satisfies Condition (4).*

The proofs of Theorems 1 and 2 are provided in Appendix A and B, respectively.

3.2. An efficient algorithm for constructing $\mathbf{X}^f$

This section presents an algorithm for constructing the finest-Condorcet partition. Beforehand, it is expedient to link the pairwise preference relationships, i.e., $s_{ij} > s_{ji} + t_{ij}$, with the elements of the CR matrix (see Section 2.1), namely to reduce storage requirements and computational effort.

Proposition 1. *Item i is pairwise preferred over item j if and only if $c_{ij} > 0$ and $c_{ji} < 0$.*

The proof is provided in Appendix C.

From Proposition 1, the CR matrix contains sufficient information to determine the pairwise preferences of all item-pairs and thereby enable Condorcet partitioning. Its use reduces storage requirements since instead of having to store $[s_{ij}] \in \mathbb{Z}^{n\times n}$ and $[t_{ij}] \in \mathbb{Z}^{n\times n}$, only $[c_{ij}] \in \mathbb{Z}^{n\times n}$ is needed. Next, we define parameters needed by the presented algorithm.

Definition 8. Let Γ_i be the set of items over which item i is pairwise preferred; its contents are given by $\Gamma_i := \{j \in \mathcal{X} : s_{ij} > s_{ji} + t_{ij}\}$, or equivalently, $\Gamma_i := \{j \in \mathcal{X} : c_{ij} > 0,\ c_{ji} < 0\}$. Additionally, let $\gamma_i := |\Gamma_i|$ denote the number of items over which i is pairwise preferred.

Definition 9. Let $\overline{\Gamma}_i := \mathcal{X}\backslash(\Gamma_i \cup \{i\})$ be the set of items over which item i is not preferred.

The following proposition serves as the foundation of the proposed algorithm, which connects the γ-values of a distinct item-pair to their relative positions in the subsets of $\mathbf{X}^f$.

Proposition 2. *If $\gamma_i > \gamma_j$, then item j cannot belong to a lower-indexed subset than item i in $\mathbf{X}^f$; additionally, if $\gamma_i = \gamma_j$, then i and j must belong to the same subset.*

Proof. We prove this by contradiction. Let $\mathbf{X}^f = \{X_1, X_2, \dots, X_w\}$ be the finest-Condorcet partition. Additionally, let item i to belong to $X_k \in \mathbf{X}^f$ and item j to belong to $X_{k'} \in \mathbf{X}^f$, where $k, k' \in \{1, \dots, w\}$, with $k < k'$. This gives that

$$\mathbf{X}^f = \{X_1, \dots, \underbrace{\{j, \dots\}}_{X_k}, \dots, \underbrace{\{i, \dots\}}_{X_{k'}}, \dots, X_w\}.$$

Letting $|X_k|$ be the number of items in subset X_k, bounds on γ_i and γ_j can be obtained as

$$|X_{k+1}| + \cdots + |X_{k'}| + \cdots + |X_w| \le \gamma_j \le |X_k| + \cdots + |X_{k'}| + \cdots + |X_w| - 1, \text{ and}$$
$$|X_{k'+1}| + \cdots + |X_w| \le \gamma_i \le |X_{k'}| + \cdots + |X_w| - 1.$$

The lower bound on γ_j comes from the definition of $\mathbf{X}^f$, since each item in X_k must be pairwise preferred over all items in subsets $X_{k+1}, \dots, X_w$. The upper bound on γ_j comes from the fact that, when $|X_k| > 1$, j can be pairwise preferred over some items in X_k, but there must be at least one item in this subset over which j is not pairwise preferred; otherwise, j must belong to X_{k-1}. Lower and upper bounds on γ_i are calculated in the same fashion. The values of γ_i and γ_j can be connected as follows:

$$\gamma_j \ge \sum_{t=k+1}^{w} |X_t| = \sum_{t=k+1}^{k'-1} |X_t| + \sum_{t=k'}^{w} |X_t| \ge \sum_{t=k'}^{w} |X_t| > \sum_{t=k'}^{w} |X_t| - 1 \ge \gamma_i.$$

Therefore, this gives that $\gamma_j > \gamma_i$, which contradicts the starting assumption. In summary, when $\gamma_i > \gamma_j$, j cannot belong to a lower-indexed subset than i in $\mathbf{X}^f$. Through a parallel chain of arguments, a similar contradiction results when $\gamma_i = \gamma_j$, meaning that i and j must belong to the same subset in the latter case. □

The pseudocode of the proposed partitioning procedure is presented in Algorithm 1, and it consists of two phases: (1) construction of an initial partition, (2) validation & merging. The algorithm utilizes Proposition 2 to build an initial partition $\mathbf{X}^0$. According to this proposition, if $\gamma_i = \gamma_j$, then i and j must belong to the same subset in $\mathbf{X}^f$; additionally, if $\gamma_i > \gamma_j$, j cannot belong to a lower-indexed subset than i in $\mathbf{X}^f$. Hence, $\{X_1^0, X_2^0, \dots, X_w^0\} = \mathbf{X}^0$ is constructed by ordering the items by non-increasing γ-values; items with a distinct value are placed in separate subsets, and items with the same value are placed in the same subset. In more detail, the item(s) with the maximum γ-value are placed in X_1^0, item(s) with the next highest value are placed in X_2^0, etc. The second phase checks whether $\mathbf{X}^0$ satisfies NXCC; if it does not, it merges the subsets that have caused the violation. This process is repeated until the working partition satisfies NXCC. The next two theorems prove that Algorithm 1 is correct, meaning that its output satisfies NXCC and condition (4), and that it has a time complexity of $O(n^2)$.

Theorem 3. *Algorithm 1 is correct.*

Theorem 4. *Algorithm 1 has a time complexity of $O(n^2)$.*

The proofs of Theorems 3 and 4 are provide in Appendix C and D, respectively.

Example 3. Consider the instance given in Example 1. The γ-values for this instance are

$$\gamma_1 = 5, \gamma_2 = 3, \gamma_3 = 3, \gamma_4 = 2, \gamma_5 = 1, \gamma_6 = 0.$$

The initial partition is $\mathbf{X}^0 = \{\{1\}, \{2, 3\}, \{4\}, \{5\}, \{6\}\}$. Here, item 1 has the highest γ-value, items 2 and 3 have the second-highest value, etc. Next, set the working partition to the initial partition (i.e., $\widetilde{\mathbf{X}} \leftarrow \mathbf{X}^0$). Afterward, start validation & merging.

Iteration 1: Item 1 is pairwise preferred over all items in the higher-indexed subsets; hence, the working partition remains unchanged.

Algorithm 1: Finest-Condorcet Partition

Input : $[c_{ij}] \in \mathbb{Z}^{n\times n}$ (CR matrix)
Output: Finest-Condorcet partition ($\mathbf{X}^f$), initial partition ($\mathbf{X}^0$)

Apply Definitions 8 and 9 to calculate parameters Γ_i, $\overline{\Gamma}_i$, and γ_i using $[c_{ij}]$, for $i \in \mathcal{X}$;
Construct the initial partition $\mathbf{X}^0$ by placing all item(s) with the highest γ-value in X_1^0, all item(s) with the next highest γ-value in X_2^0, etc.;
$\widetilde{\mathbf{X}} \leftarrow \{X_1^0, X_2^0, \ldots, X_w^0\} = \mathbf{X}^0$; // set working partition to initial partition
$k = 1$;
while $k < |\boldsymbol{X}^0|$ **do**
  $\overline{\Gamma}(\widetilde{X}_k) \leftarrow \cup_{i\in\widetilde{X}_k} \overline{\Gamma}_i$; // get items over which items in $\widetilde{X}_k$ are not pairwise preferred
  if $\overline{\Gamma}(\widetilde{X}_k) \setminus \cup_{t=1}^{k} \widetilde{X}_t = \emptyset$; // if $\widetilde{X}_k$ does not violate NXCC **then**
    $k \leftarrow k + 1$;
  else
    while $\overline{\Gamma}(\widetilde{X}_k) \setminus \cup_{t=1}^{k} \widetilde{X}_t \neq \emptyset$ **do** ; // while subset $\widetilde{X}_k$ violates NXCC
      $k' \leftarrow \max\{g : i \in \overline{\Gamma}(\widetilde{X}_k) \setminus \cup_{t=1}^{k} \widetilde{X}_t \wedge i \in \widetilde{X}_g\}$; // get max-index subset where a violation is detected
      $\widetilde{X}_k \leftarrow \cup_{t=k}^{k'} \widetilde{X}_t$; // merge subsets causing NXCC violation into $\widetilde{X}_k$
    $k \leftarrow k' + 1$;
$\mathbf{X}^f \leftarrow \widetilde{\mathbf{X}}$;
return $\boldsymbol{X}^f, \boldsymbol{X}^0$

Iteration 2: Item 2 is not pairwise preferred over item 4. Hence, subsets $\{2,3\}$ and $\{4\}$ are merged to satisfy NXCC. This gives that $\widetilde{\mathbf{X}} = \{\{1\},\{2,3,4\},\{5\},\{6\}\}$.
Iteration 3: Items 2, 3, 4 are pairwise preferred over items 5 and 6; hence, the working partition remains unchanged.
Iteration 4: Item 5 is pairwise preferred over item 6. Therefore, $\mathbf{X}^f = \{\{1\},\{2,3,4\},\{5\},\{6\}\}$.

4. Approximate Condorcet partitioning

Condorcet partitioning can be very useful for expediting Kemeny-Agg, particularly when the resulting partition has many small subsets. However, some instances are not partitionable, and, in various other cases when they are, the partition may yield relatively few subsets and/or very large subsets. Yoo and Escobedo (2021) reported that a sizeable fraction of the real-world instances with ties drawn from the Preflib data set (Mattei and Walsh, 2013) yielded Condorcet partitions with these disadvantageous characteristics: Nearly 40% of the tested instances, which contained up to 300 items, were not partitionable at all. Betzler et al. (2014) reported similar results on synthetic instances generated via the Plackett–Luce model (Luce, 2012; Plackett, 1975): Out of four synthetic instances, the two largest instances with 100 and 200 items, respectively, had Condorcet partitions with one subset containing over 95% of the items. Pilot experiments conducted herein yielded similar results: 109 out of 113 instances with more than 100 items drawn from three different real-world data sets were either not partitionable or they had a subset that contained more than 95% of the items. Such results indicate that exact decomposition is useful only for a limited number of instances. This section introduces Approximate Condorcet Partitioning (ACP), which can be applied to any strict or non-strict instance of Kemeny-Agg. Whenever NXCC yields a partition with at least one large subset, ACP leverages both the finest-Condorcet partition and the initial partition constructed from γ-values to return a partition with relatively more and smaller subsets, which is not strictly a Condorcet partition (i.e., the subset orderings may conflict with the Kemeny optimal solution(s)) but retains some of the computationally beneficial structure of this social choice-inspired concept. ACP is a scalable solution technique for solving Kemeny-Agg. Formal guarantees of the resulting solutions are also derived later in this section.

Finally, an important remark regarding the constant-factor approximation schemes for Kemeny-Agg is in order. Except for Spearman's footrule, the approximation algorithms introduced in Section 1 do not guarantee their respective approximation factor over all instances. Rather, their guarantees are achieved *on average*. We illustrate the potential for high variability in solution quality of such expected approximation algorithms using Pick-A-Perm. Let $\boldsymbol{\sigma}^1 = \cdots = \boldsymbol{\sigma}^9 = [1,2,3]^T$ and $\boldsymbol{\sigma}^{10} = [1,3,2]^T$. Here, $\boldsymbol{\sigma}^* = \boldsymbol{\sigma}^1$, with a cumulative Kemeny–Snell distance to the input rankings of 2. However, Pick-A-Perm may still choose $\boldsymbol{\sigma}^{10}$ (with a probability of 1/10), which has a cumulative Kemeny–Snell distance to the input rankings of 18 (9-times the expected factor). The rest of this section is organized as follows. Section 4.1 develops ACP, and Section 4.2 derives solution guarantees for any item-partitioning scheme and improved guarantees for ACP.

4.1. Applying ACP to solve Kemeny-Agg

Before proceeding, it is important to emphasize that ACP is primarily designed for solving large-scale problems (instances with thousands of items), where state-of-the-art algorithms like LPKwikSort are not practical or where the solution guarantees of existing approximation algorithms are not satisfactory.

Whenever $\mathbf{X}^f$ contains one or more large subsets, ACP constructs a partition of $\mathcal{X}$ that leverages the finest-Condorcet partition, $\mathbf{X}^f$, and the initial partition, $\mathbf{X}^0$, both obtained from Algorithm 1. Recall that $\mathbf{X}^0$ is easily constructed based on the calculated parameter γ_i, defined as the number of items over which item i is pairwise preferred. Typically, $\mathbf{X}^0$ consists of many subsets, a large fraction of which are subsequently merged in Algorithm 1 to satisfy NXCC. Whenever the validation & merging step creates large subsets in $\mathbf{X}^f$, ACP builds a different item partition from $\mathbf{X}^0$, which may violate NXCC but retains some of the convenient structure of $\mathbf{X}^f$. That is, only those subsets of $\mathbf{X}^f$ which are difficult to solve by exact methods are broken down by ACP; all other subsets are left unchanged, and this preserves some of the ordered item-subsets. Thus, this new partition is designed to yield a higher number of computationally manageable subsets, i.e., whose Kemeny-Agg subproblems are solvable with exact methods. A key insight behind ACP is that, items that have close γ-values are more likely to be close to each other in the aggregate ranking; hence, smaller subsets are formed based on these calculated parameters, keeping those with similar values near one another. The pseudocode of the proposed algorithm for ACP is presented in Algorithm 2.

To summarize its steps, let h be a user-specified threshold, which can be set to the maximum Kemeny-Agg instance size that is solvable to optimality within a reasonable time (i.e., based on prior findings and available computational resources). That is, for some subset $X_k^f \in \mathbf{X}^f$ with $|X_k^f| \leq h$, ACP skips this subset. However, if $|X_k^f| > h$, then the algorithm evaluates the corresponding adjacent subsets in $\mathbf{X}^0$, say $\{X_u^0, \ldots, X_v^0\} = X_{uv}^0$, which were merged together during the validation & merging step of Algorithm 1 to form X_k^f. In the next step, the algorithm tries to merge adjacent subsets of X_{uv}^0 as long as their combined size does not exceed h. During this process, if the size of any individual subset comprising X_{uv}^0 is already greater than h, that subset is not merged with any other subsets and is left unchanged for the remainder of the algorithm.

Let $\mathbf{X}_h^{ACP}$ denote the partition obtained using threshold h. After obtaining ACP, the ensuing steps aim to obtain a high quality solution via ACP: (1) solve those subsets of $\mathbf{X}_h^{ACP}$ whose size is at most h to optimality, (2) for each subset whose size exceeds h, tie all its items in the case of non-strict rankings and permute its items randomly in the case of strict rankings. Step 2 aims to find a quick solution for those subset that are deemed difficult to solve to optimality. Similar

Algorithm 2: Approximate Condorcet Partitioning

Input : $[c_{ij}] \in \mathbb{Z}^{n\times n}$ (CR matrix), h
Output: Approximate Condorcet Partition ($\mathbf{X}_h^{ACP}$)

$\mathbf{X}^f$, $\mathbf{X}^0 \leftarrow$ Finest-Condorcet Partition($[c_{ij}]$);
$\mathbf{X}_h^{ACP} \leftarrow \emptyset$;
for $k = 1$ **to** $|\mathbf{X}^f|$ **do**
  if $|X_k^f| \le h$ **then**
    Append $\mathbf{X}_k^f$ to $\mathbf{X}_h^{ACP}$;
  else
    Let $\{X_u^0, \dots, X_v^0\} = X_{uv}^0$ be the set of consecutive subsets of $\mathbf{X}^0$ that have been merged together to form $\mathbf{X}_k^f$;
    $q \leftarrow u$;
    while $q \le v - 1$ **do**
      if $|X_q^0| \ge h$ ***or*** $q = v - 1$ **then**
        Append X_q^0 to $\mathbf{X}_h^{ACP}$;
        $q \leftarrow q + 1$;
      else
        Let $l \le v$ be the highest index such that $|X_q^0 \cup \dots \cup X_l^0| \le h$;
        Merge subsets $X_q^0, \dots, X_l^0$ and append it to $\mathbf{X}_h^{ACP}$;
        $q \leftarrow l + 1$;
return $\mathbf{X}_h^{ACP}$

to Condorcet partitioning, to obtain a complete ordering of $\mathcal{X}$, items in lower-indexed subsets of $\mathbf{X}_h^{ACP}$ are strictly ranked ahead of items in higher-indexed subsets.

Remark 1. The proposed solution method via ACP becomes an exact method if $|X_k^f| \le h \ \forall X_k \in \mathbf{X}^f$.

The ensuing small example helps illustrate ACP.

Example 4. Consider an instance with 5 rankings of 10 items and set the threshold as $h = 3$. The input rankings ($\boldsymbol{\sigma}^1, \dots, \boldsymbol{\sigma}^5$), aggregate ranking ($\boldsymbol{\sigma}^*$), and γ-value of each item are given by

Item	Input rankings					$\boldsymbol{\sigma}^*$
	$\boldsymbol{\sigma}^1$	$\boldsymbol{\sigma}^2$	$\boldsymbol{\sigma}^3$	$\boldsymbol{\sigma}^4$	$\boldsymbol{\sigma}^5$	
1	8	1	9	10	1	7
2	2	3	7	6	8	4
3	5	10	2	4	10	5
4	6	5	5	2	2	2
5	3	4	10	9	9	10
6	1	9	1	1	4	1
7	10	7	6	5	5	9
8	9	6	4	8	7	8
9	7	2	8	7	3	6
10	4	8	3	3	6	3

Item	γ
1	3
2	5
3	4
4	8
5	1
6	9
7	2
8	2
9	4
10	7

The initial partition is $\mathbf{X}^0 = \{\{6\},\{4\},\{10\},\{2\},\{3,9\},\{1\},\{7,8\},\{5\}\}$, and the finest-Condorcet partition is $\mathbf{X}^f = \{\{6\},\{4\},\{10\},\{1,2,3,5,7,8,9\}\}$. The ACP algorithm leaves subsets $\{6\},\{4\}$ and $\{10\}$ unchanged, as their sizes are less than h, but it seeks to further decompose the fourth subset of $\mathbf{X}^f$ whose size exceeds h. Note that, subsets $\{2\},\{3,9\},\{1\},\{7,8\},\{5\} \in \mathbf{X}^0$ were merged in the validation & merging to form subset $\{1,2,3,5,7,8,9\}$. ACP proceeds by merging subsets $\{2\},\{3,9\}$ to form subset $\{2,3,9\}$ whose size reaches h; subsets $\{1\},\{7,8\}$ are merged to form subset $\{1,7,8\}$ whose size also reaches the threshold; and subset $\{5\}$ is left unchanged. Therefore, the output of ACP is given by $\mathbf{X}_{h=3}^{ACP} = \{\{6\},\{4\},\{10\},\{2,3,9\},\{1,7,8\},\{5\}\}$. Afterward, one Kemeny-Agg subproblem is solved for each subset and their respective solutions are concatenated (for completeness, the concatenated subproblem solutions matches the optimal solution of the full problem, for this example).

4.2. Provable guarantees from partitioning

This subsection derives three different approximation factors, all of which are easy to calculate and specific to the characteristics of an instance. The first of these is applicable to any item-partitioning scheme, including those that may not be consistent with Condorcet properties — e.g., see Aledo et al. (2021) for a decomposition based on Borda scores and Liu et al. (2021) for a hierarchical clustering method. The second and third derived approximation factors provide improved guarantees for the ACP solution, for non-strict and strict ranking instances, respectively.

For the remainder of this section, let $d_{KS}(\boldsymbol{\sigma}^*)$ denote the cumulative Kemeny–Snell distance of the aggregate ranking $\boldsymbol{\sigma}^*$ to all input rankings, and LB be a lower bound on $d_{KS}(\boldsymbol{\sigma}^*)$.

Lemma 1. *Let $\mathbf{X} = \{X_1, X_2, \dots, X_w\}$ be any given partition of X and $\hat{\boldsymbol{\sigma}}$ be a complete ranking obtained by independently solving the subsets of $\mathbf{X}$ and concatenating the solutions of these subproblems. If $d_{KS}(\hat{\boldsymbol{\sigma}}) - d_{KS}(\boldsymbol{\sigma}^*)$ is bounded by a constant β, the complete ranking $\hat{\boldsymbol{\sigma}}$ is an $(1+\alpha)$-approximate solution, where $\alpha = \beta / LB$.*

Proof.

$$d_{KS}(\hat{\boldsymbol{\sigma}}) \le d_{KS}(\boldsymbol{\sigma}^*) + \beta = d_{KS}(\boldsymbol{\sigma}^*) + \alpha LB \le d_{KS}(\boldsymbol{\sigma}^*) + \alpha d_{KS}(\boldsymbol{\sigma}^*) = (1+\alpha) d_{KS}(\boldsymbol{\sigma}^*).$$

□

Lemma 2. *Let $\mathbf{X} = \{X_1, X_2, \dots, X_w\}$ be any given partition of $\mathcal{X}$ and $\hat{\boldsymbol{\sigma}}$ be a complete ranking obtained by independently solving the subsets of $\mathbf{X}$ (using any method of choice) and concatenating the solutions of these subproblems. The term $d_{KS}(\hat{\boldsymbol{\sigma}}) - d_{KS}(\boldsymbol{\sigma}^*)$ is bounded by*

$$2\sum_{k=1}^{w-1}\sum_{k=k+1}^{w}\sum_{i\in X_k}\sum_{j\in X_{k'}} \max\big(0, (2s_{ji}+t_{ij}) - (2s_{ij}+t_{ij}), (2s_{ji}+t_{ij}) - (s_{ij}+s_{ji})\big) + \sum_{k=1}^{w}\sum_{(i,j)\in X_k} \max\big(2s_{ij}+t_{ij}, 2s_{ji}+t_{ij}, s_{ij}+s_{ji}\big) - \min\big(2s_{ij}+t_{ij}, 2s_{ji}+t_{ij}, s_{ij}+s_{ji}\big). \tag{5}$$

Combining Lemmas 1 and 2 provides a formal guarantee of the solution quality of an arbitrary partition $\mathbf{X}$. The approximation factor holds regardless of how the items within each subset in the partition are ordered (i.e., it is a worst-case bound), and thus any method of choice can be used. As such, the quality of the solution is improved by determining orderings that more closely align with the optimal solution. The next two theorems derive a tighter guarantee by leveraging the specific solution methods for solving the subsets of $\mathbf{X}_h^{ACP}$.

Theorem 5. *Assume that the input rankings are non-strict and let $\mathbf{X}_h^{ACP} = \{X_1, \dots, X_w\}$ be the ACP partition obtained using threshold h. Let $\hat{\boldsymbol{\sigma}}$ be the complete ranking obtained via ACP from the following two steps: (1) solve subsets of at most size h to optimality, (2) tie all items in subsets of size greater than h. The term $d_{KS}(\hat{\boldsymbol{\sigma}}) - d_{KS}(\boldsymbol{\sigma}^*)$ is bounded by*

$$2\sum_{k=1}^{w-1}\sum_{k=k+1}^{w}\sum_{i\in X_k}\sum_{j\in X_{k'}} \max\big(0, (2s_{ji}+t_{ij}) - (2s_{ij}+t_{ij}), (2s_{ji}+t_{ij}) - (s_{ij}+s_{ji})\big) + \sum_{k=1}^{w}\sum_{(i,j)\in X_k} \big((s_{ij}+s_{ji}) - \min\big(2s_{ij}+t_{ij}, 2s_{ji}+t_{ij}, s_{ij}+s_{ji}\big)\big). \tag{6}$$

Theorem 6. *Assume that the input rankings are strict and let $\mathbf{X}_h^{ACP} = \{X_1, \dots, X_w\}$ be the ACP partition obtained using threshold h. Let $\hat{\boldsymbol{\sigma}}$ be the complete ranking obtained via ACP from the following two steps: (1) solve*

subsets of at most size h to optimality, (2) randomly permute items within subsets of size greater than h. The term $d_{KS}(\hat{\boldsymbol{\sigma}}) - d_{KS}(\boldsymbol{\sigma}^*)$ *is bounded by*

$$4\sum_{k=1}^{w-1}\sum_{k=k+1}^{w}\sum_{i\in X_k}\sum_{j\in X_{k'}}\max\left(0,\, s_{ji}-s_{ij}\right) + 2\sum_{k=1}^{w}\sum_{(i,j)\in X_k}\left(s_{ji}\mathbb{1}_{\overline{\sigma}_i<\overline{\sigma}_j} + s_{ij}\mathbb{1}_{\overline{\sigma}_j<\overline{\sigma}_i} - \min\left(s_{ij},\, s_{ji}\right)\right); \tag{7}$$

where $\overline{\boldsymbol{\sigma}}$ *is an auxiliary ranking obtained by randomly permuting all items in subset* X_k, $\forall k \in \{1,\ldots,w\}$ *(while ranking items in the lower-indexed subsets strictly ahead of items in the higher-indexed subsets), and* $\mathbb{1}$ *is an indicator function.*

The approximation factors are computed after a partition is obtained, meaning they are instance-specific and not constant; their value becomes relatively small when the given partition aligns well with the structure of the aggregate ranking. ACP offers significant advantages over various other partitioning methods in this regard. Furthermore, it uses the calculated γ-parameters used to obtain $\mathbf{X}^0$ to reduce the number of rank reversals between items across many more subsets than are contained in $\mathbf{X}^f$. It also leverages structural information from $\mathbf{X}^f$ by retaining item-ordering of subsets that are relatively easy to solve and their relative ordering to other subsets. It is important to emphasize that, while the derived approximation factors provide a guarantee of the solution quality, we are interested in partitions that tend to produce high quality solutions rather than those that minimize the approximation factor. Indeed, by increasing the number of subsets (i.e., reducing the value of h), one may decrease their values; however, doing so can also negatively impact the resulting solution, as increasing the number of subsets can be viewed as placing more constraints on the ordering of certain items.

The introduced approximation factors require a lower bound on $d_{KS}(\boldsymbol{\sigma}^*)$. A lower bound on $d_{KS}(\boldsymbol{\sigma}^*)$ is given by (Akbari and Escobedo, 2021):

$$LB = 2\sum_{i,j\in X}\min\left(2s_{ji}+t_{ij},\, 2s_{ij}+t_{ij},\, s_{ij}+s_{ji}\right). \tag{8}$$

The lower bound defined in Eq. (8) equals zero if and only if all input rankings are identical, which renders the approximation factors incomputable; however, this special case does not require analysis of any kind (i.e., the aggregate ranking equals the unanimous ranking), meaning that it does not pose a serious issue for the proposed approximation algorithm. We reckon that there are other lower bounds for the case of strict rankings (Conitzer et al., 2006) and non-strict rankings (Akbari and Escobedo, 2021). The lower bound defined in Eq. (8) has been utilized as it is very fast to compute (Akbari and Escobedo, 2021).

5. Computational comparisons

This section compares ACP with some of the prominent approximation schemes mentioned in Section 1. The selected methods for instances with strict rankings are the proposed solution method via ACP, BestInput, DeterministicKwikSort, KwikSort, LPKwikSort, and Spearman's footrule. For instances with non-strict rankings, we elected not to use RepeatChoice and LPKwikSort$_h$ (Ailon, 2010) as they restrict the aggregate ranking to be strict, which does not align with the more general assumption that the output rankings may also be non-strict. Due to a lack of suitable algorithms (and to compare ACP with more than simply BestInput) for Kemeny-Agg with non-strict rankings, we modify KwikSort to handle these instances and denote the resulting algorithm as NonStrictKwikSort; the pseudocode and its description are presented in Appendix I.

We use two real-world data sets. The first data set is from Cohen-Boulakia et al. (2011) and is henceforth denoted as the Biomedical data set. Each instance of this data set contains four non-strict rankings of genes possibly associated with Breast Cancer, Prostate Cancer, Bladder Cancer, Neuroblastoma, Retinoblastoma, ADHD (Attention Deficit Hyperactivity Disorder), and LQTS (Long QT Syndrome). Each set of input rankings is the result of querying for the respective diseases in biological databases using four different methods. The objective of the referenced study is to reduce the variability of information retrieval techniques by consolidating their outputs. The second data set consists of instances with and without ties from Preflib (Mattei and Walsh, 2013), a library of preference data; namely instances from "TOC - Orders with Ties - Complete List" and "SOC - Strict Orders - Complete List" with over 100 items.[1]

All experiments were carried out on a PC with an Intel(R) Xeon(R) CPU E5-2680 @2.40 GHz with 64 GB RAM. All Kemeny-Agg subproblems were solved using the exact binary programming formulation of Yoo and Escobedo (2021) via CPLEX solver version 12.10.0. The Spearman's footrule rank aggregation problem was solved via minimum cost perfect matching in bipartite graphs (Dwork et al., 2001). For ACP, we tested thresholds $h = 30, 40, 50$. The experimental results report, for each instance, number of items (n), number of input rankings (m), size of the largest subset of $\mathbf{X}^f$ ($(X_l^f)^{max}$), run-time (Time) and relative optimality gap (Gap %) attained by each tested method. Run-times include pre-processing time required by each specific method. The relative optimality gap for each method is calculated as the difference between its objective value and the lower bound (Eq. (8)), divided by the lower bound. It is displayed as a percentage, for convenience (relative optimality gap, multiplied by 100); the best relative optimality gap % attained for each instance is shown in bold. For ACP, the tested threshold value (h) and calculated approximation factor (AF) are reported. Lastly, for each data set, the average relative optimality gap and geometric mean of run-times achieved by each of the selected algorithms are displayed. All statistics are rounded to two decimal points, and all reported run times are in seconds. For completeness, the objective functions values of the tested algorithms — that is, the cumulative d_{KS} distance of their solution ranking to all the input rankings — are reported in Appendix J.

Table 1 reports the computational results for the Biomedical data set. These results exclude the ADHD and LSQT instances since they were both relatively small and could be easily solved to optimality without partitioning. Compare table columns n and $(X_l^f)^{max}$ (the number of items and the size of the largest subset of $\mathbf{X}^f$) to observe that the finest-Condorcet partition over all instances is either null or it contains a rather large subset that is difficult to solve to optimality. Overall, none of the selected methods had a dominant performance on all of these instances. On average, BestInput had the lowest relative optimality gap, ACP the second-lowest, and NonStrictKwikSort the third-lowest. As Table 1 shows, BestInput and ACP had a top-2 performance in terms of solution quality for all five instances. Among the three tested threshold values for ACP, $h = 50$ achieved the best solution quality, but it also had higher run-times and approximation factors. The average approximation factor it achieved over these instances was 1.14. Finally, ACP yielded near-optimal solutions (with gaps of up to 0.11%) on Bladder Cancer and Retinoblastoma, for which each $\mathbf{X}^f$ contained multiple subsets of small-to-medium sizes.

Table 2 reports the results of the Preflib TOC data set. The general characteristics of $\mathbf{X}^f$ were the same as for the Biomedical data set. Overall, ACP exhibited a dominant performance, achieving the lowest relative optimality gap for all 85 instances. In fact, its worst optimality gap of 3.49% over all instances and three tested thresholds was lower than all optimality gaps achieved by BestInput. Most impressively, its average approximation factor was 1.02, further highlighting the comparative robustness of ACP. To round out the results, NonStrictKwikSort was faster but yielded a slightly lower solution quality than BestInput

[1] The Preflib data set presents the input rankings in the form of sorted lists, but a few items repeat in certain instances, presumably due to error. To overcome this issue, we adjusted these instances by keeping the first appearance of each item in each list and deleting any extra occurrences

Table 1
Performance metrics of the selected algorithms for solving instances of the Biomedical data set; times are reported in seconds (s).

Instance	n	m	$(X_l^f)^{max}$	NonStrictKwikSort		BestInput		Approximate Condorcet Partitioning								
								$h = 30$			$h = 40$			$h = 50$		
				Gap %	Time (s)	Gap %	Time (s)	Gap %	Time (s)	AF	Gap %	Time (s)	AF	Gap %	Time (s)	AF
Prostate Cancer	218	4	216	41.1	0.05	**12.8**	0.21	22.99	1.07	1.23	22.4	1.92	1.22	22.34	2.28	1.22
Bladder Cancer	308	4	266	16.54	0.11	**0.00**	0.45	0.01	0.95	1.01	0.01	0.85	1.01	0.01	2.19	1.01
Breast Cancer	386	4	386	34.97	0.18	**5.32**	0.67	29.12	2.72	1.29	29.32	4.78	1.29	29.52	6.39	1.3
Retinoblastoma	402	4	358	0.88	0.18	0.19	0.74	**0.11**	0.51	1.01	**0.11**	1.6	1.01	**0.11**	2.55	1.01
Neuroblastoma	431	4	431	15.78	0.18	5.43	0.82	4.23	1.4	1.05	3.91	4.55	1.04	**3.76**	7.74	1.04
Average				21.85	0.13	4.75	0.52	11.29	1.15	1.12	11.15	2.24	1.12	11.15	3.63	1.14

Table 2
Performance metrics of the selected algorithms for solving instances of the TOC data set with more than 100 items; times are reported in seconds (s).

Instance	n	m	$(X_l^f)^{max}$	NonStrictKwikSort		BestInput		Approximate Condorcet Partitioning								
								$h = 30$			$h = 40$			$h = 50$		
				(%) Gap	Time (s)	(%) Gap	Time (s)	(%) Gap	Time (s)	AF	(%) Gap	Time	AF	(%) Gap	Time (s)	AF
ED-14-02	100	5000	100	0	2.48	–	≥ 3600	**0.00**	2.5	1.00	**0.00**	2.5	1.00	**0.00**	2.5	1.00
ED-14-03	100	5000	100	0	2.17	–	≥ 3600	**0.00**	2.17	1.00	**0.00**	2.17	1.00	**0.00**	2.19	1.00
MD-03-03	102	32	102	0	0.01	52.07	4.07	**0.00**	0.02	1.00	**0.00**	0.02	1.00	**0.00**	0.02	1.00
MD-03-05	103	31	103	0	0.02	69.32	4.85	**0.00**	0.02	1.00	**0.00**	0.02	1.00	**0.00**	0.02	1.00
MD-03-06	133	38	133	0	0.02	69.72	11.56	**0.00**	0.02	1.00	**0.00**	0.02	1.00	**0.00**	0.02	1.00
MD-03-08	147	51	147	0	0.02	57.58	25.17	**0.00**	0.03	1.00	**0.00**	0.03	1.00	**0.00**	0.05	1.00
MD-03-07	155	51	155	0	0.03	69.72	20.96	**0.00**	0.06	1.00	**0.00**	0.06	1.00	**0.00**	0.06	1.00
ED-10-50	170	4	170	36.63	0.02	14.17	0.14	**1.44**	1.48	1.07	1.57	1.86	1.10	1.47	3.2	1.13
ED-10-49	351	4	351	15.39	0.30	16.58	0.77	0.75	2.85	1.03	0.75	4.19	1.04	**0.65**	7.32	1.04
ED-18-01	379	723	379	99.82	0.41	–	≥ 3600	**0.00**	0.44	1.00	**0.00**	0.45	1.00	**0.00**	0.42	1.00
ED-18-03	477	556	476	99.36	0.45	–	≥ 3600	**0.00**	0.48	1.00	**0.00**	0.48	1.00	**0.00**	0.47	1.00
ED-11-12	1210	4	1207	11.17	1.27	4.93	6.36	2.81	6.13	1.03	2.79	9.24	1.03	**2.77**	13.44	1.03
ED-11-31	1223	4	1223	16.69	1.16	8.78	6.58	1.73	5.77	1.02	1.72	8.63	1.02	**1.69**	12.24	1.02
ED-11-09	1272	4	1272	22.55	1.52	8.45	6.85	3.49	8.59	1.04	3.46	13.83	1.04	**3.39**	20.90	1.04
ED-11-23	1342	4	1341	10.96	1.39	9.39	7.98	1.68	6.77	1.02	1.65	10.72	1.02	**1.62**	15.36	1.02
ED-11-21	1347	4	1347	10.28	1.56	5.24	7.69	2.71	8.16	1.03	2.68	12.47	1.03	**2.66**	19.13	1.03
ED-11-37	1351	4	1351	16.4	1.66	4.53	8.11	3.15	9.86	1.04	3.08	15.83	1.04	**3.04**	24.15	1.04
ED-11-25	1356	4	1353	18.23	1.55	8.78	7.89	2.39	8.56	1.03	2.31	14.05	1.03	**2.27**	19.42	1.03
ED-11-13	1363	4	1363	5.03	1.47	6.59	8.2	1.63	6.44	1.02	1.61	9.38	1.02	**1.6**	13.38	1.02
ED-11-29	1368	4	1368	25.91	1.75	4.08	8.55	3.4	10.20	1.04	3.37	16.67	1.04	**3.31**	25.94	1.04
ED-11-14	1375	4	1372	27.29	1.77	3.87	8.36	3.32	9.89	1.04	3.24	16.03	1.04	**3.2**	23.74	1.04
ED-11-30	1386	4	1384	4.96	1.64	6.26	8.78	2.46	8.85	1.03	2.41	14.41	1.03	**2.38**	21.21	1.03
ED-11-06	1449	4	1449	17.69	1.77	5.22	10.02	2.8	9.69	1.03	2.75	14.52	1.03	**2.71**	22.66	1.03
ED-11-04	1467	4	1463	48.04	1.91	4.94	10.63	2.64	8.85	1.03	2.61	13.3	1.03	**2.59**	19.86	1.03
ED-11-07	1474	4	1470	3.72	1.73	8.57	10.19	1.84	7.3	1.02	1.82	10.61	1.02	**1.81**	15.7	1.02
ED-11-34	1509	4	1509	12.92	1.77	6.94	10.02	1.54	7.36	1.03	1.51	10.68	1.03	**1.5**	14.91	1.03
ED-11-22	1514	4	1513	18.74	1.86	5.92	10.09	1.9	9.52	1.02	1.85	14.7	1.02	**1.83**	20.88	1.02
ED-11-11	1545	4	1542	38.89	2.08	5.75	10.36	2.1	10.91	1.02	2.04	17.04	1.02	**2.01**	25.27	1.03
ED-11-15	1563	4	1560	38.81	1.97	6.44	10.74	2.06	9.36	1.02	2.03	14.82	1.02	**2.01**	20.75	1.02
ED-11-08	1572	4	1569	7.45	1.94	6.8	11.73	1.28	6.75	1.02	1.27	9.31	1.02	**1.26**	13.25	1.02
ED-11-28	1616	4	1611	2.76	1.89	14.38	12.05	0.79	6.72	1.01	**0.78**	9.0	1.01	**0.78**	12.38	1.01
ED-11-40	1623	4	1623	35.54	1.59	19.18	11.88	0.33	6.43	1.01	**0.32**	10.49	1.01	**0.32**	11.64	1.02
ED-11-36	1634	4	1632	28.92	2.06	8.13	11.85	1.55	9.03	1.02	1.52	13.19	1.02	**1.51**	18.36	1.02
ED-11-33	1646	4	1644	9.96	2.3	5.82	12.5	1.94	10.35	1.02	1.9	15.31	1.02	**1.85**	22.41	1.02
ED-11-05	1673	4	1672	4.07	1.98	23.32	13.52	0.66	9.91	1.01	0.64	17.89	1.01	**0.63**	88.36	1.01
ED-11-18	1681	4	1676	12.33	2.22	6.24	12.14	1.98	8.85	1.02	1.97	13.56	1.02	**1.94**	18.66	1.02
ED-11-16	1708	4	1707	6.93	2.19	8.07	13.07	1.19	8.73	1.02	**1.17**	12.64	1.02	**1.17**	17.61	1.02
ED-11-32	1751	4	1751	9.45	2.3	5.57	13.1	1.71	8.92	1.02	**1.68**	12.56	1.02	**1.68**	17.69	1.02
ED-11-38	1754	4	1752	13.8	2.36	6.03	14.16	2.02	9.06	1.02	1.99	12.83	1.02	**1.97**	17.92	1.02
ED-11-39	1788	4	1788	35.34	2.2	18.19	13.75	0.39	9.22	1.01	0.37	18.8	1.01	**0.35**	24.71	1.01

(*continued on next page*)

on average. NonStrictKwikSort had an optimality gap of up to 97.92% and BestInput of up to 69.72%. BestInput did not terminate after 1 h of run-time for instances #77–#80, likely owing to its quadratic complexity with respect to both the number of items and number of input rankings. These four instances are much larger than the rest: #77–#78 have 5,000 input rankings and #79–#80 have at least 379 items and at least 556 input rankings.

Table 3 reports the results of the Preflib SOC data set. The general characteristics of $\mathbf{X}^f$ were the same as for the TOC and the Biomedical data sets. Since all of the alternative methods tested for this data set output a strict ranking, the output ranking of ACP was forced to be strict as well. While this restriction does not take full advantage its intended purpose, ACP still exhibited a very good performance, headlined by its average approximation factor of 1.06 over this data set. While LPKwikSort dominated in solution quality, achieving the lowest optimality gap in all but one of the 23 tested instances, it also had relatively high run-times — in fact, its lowest run-time was greater than the highest run-time attained by all other methods. This is due to fact that LPKwikSort requires solving a LP with $O(n^3)$ constraints, which causes memory issues for large instances. Conversely, ACP produced competitive solutions in far less time (it solved each instance of the SOC data set in under eight seconds). In the 22 instances where LPKwikSort had the lowest optimality gap, ACP had the second-lowest in 14 instances and BestInput in 8 instances; however, the worst relative optimality gap of BestInput (14.94%) was much higher than ACP's (not more than 4.07%). Furthermore, the relative optimality gaps attained

Table 2 (*continued*).

Instance	n	m	$(X_i^f)^{max}$	NonStrictKwikSort		BestInput		Approximate Condorcet Partitioning								
								$h = 30$			$h = 40$			$h = 50$		
				(%) Gap	Time (s)	(%) Gap	Time (s)	(%) Gap	Time (s)	AF	(%) Gap	Time	AF	(%) Gap	Time (s)	AF
ED-11-68	1826	4	1826	45.79	2.14	10.22	14.36	**0.62**	6.3	1.01	**0.62**	7.74	1.01	**0.62**	9.11	1.01
ED-11-49	1845	4	1844	2.94	2.6	6.7	14.74	0.91	6.56	1.01	**0.9**	8.1	1.01	**0.9**	9.17	1.01
ED-11-20	1870	4	1866	3.0	2.77	7.52	15.13	1.55	11.7	1.02	1.51	17.33	1.02	**1.47**	24.35	1.02
ED-11-26	1931	4	1930	7.41	2.80	6.36	16.63	1.39	11.03	1.02	1.36	16.85	1.02	**1.33**	23.24	1.02
ED-11-35	1936	4	1935	17.19	2.80	6.25	16.69	1.58	11.02	1.02	1.56	15.58	1.02	**1.53**	21.99	1.02
ED-11-74	1976	4	1976	5.74	2.92	5.93	17.89	1.16	9.91	1.02	1.14	13.7	1.02	**1.13**	18.69	1.02
ED-11-60	1977	4	1976	3.7	2.69	10.98	16.94	**0.87**	7.94	1.01	**0.87**	10.22	1.01	**0.87**	12.66	1.01
ED-11-58	2011	4	2010	3.43	2.94	7.26	17.28	1.1	8.95	1.02	**1.09**	11.77	1.02	**1.09**	15.36	1.02
ED-11-62	2014	4	2013	4.04	2.86	12.49	17.50	0.89	9.53	1.01	0.87	12.7	1.01	**0.86**	16.69	1.01
ED-11-17	2015	4	2014	14.86	3.08	6.03	17.68	1.31	11.49	1.03	1.28	16.78	1.03	**1.26**	22.6	1.03
ED-11-66	2024	4	2024	7.68	3.22	5.11	18.31	2.01	11.64	1.02	1.99	16.32	1.03	**1.98**	22.28	1.03
ED-11-24	2049	4	2049	6.29	3.11	5.75	18.28	1.44	11.92	1.03	1.42	16.83	1.03	**1.41**	23.02	1.03
ED-11-67	2066	4	2066	7.65	3.16	7.54	18.43	1.31	10.17	1.02	**1.3**	14.09	1.03	**1.3**	19.19	1.02
ED-11-27	2092	4	2088	22.28	3.11	7.32	19.39	1.13	9.92	1.02	**1.11**	13.46	1.02	**1.11**	17.97	1.02
ED-11-10	2096	4	2095	6.18	2.95	13.64	19.08	0.65	10.53	1.01	0.63	13.88	1.01	**0.62**	19.52	1.01
ED-11-19	2104	4	2102	10.51	3.25	6.56	19.24	1.28	12.38	1.02	1.26	17.78	1.02	**1.24**	25.38	1.02
ED-11-50	2111	4	2111	1.82	3.01	8.19	20.30	0.62	7.95	1.01	**0.61**	9.12	1.01	**0.61**	10.89	1.01
ED-11-51	2112	4	2112	6.8	3.28	4.92	19.6	1.54	10.81	1.02	1.52	14.85	1.02	**1.51**	19.92	1.02
ED-11-65	2119	4	2118	3.12	2.84	21.21	19.91	0.28	7.30	1.00	**0.27**	8.92	1.00	0.28	9.66	1.00
ED-11-41	2123	4	2123	8.44	3.28	7.5	19.6	1.05	9.03	1.01	**1.04**	11.45	1.01	**1.04**	14.25	1.02
ED-11-71	2127	4	2127	18.27	3.13	8.96	19.55	0.52	8.10	1.01	**0.51**	9.41	1.01	**0.51**	12.25	1.01
ED-11-46	2133	4	2133	4.83	3.22	10.31	20.11	1.37	9.96	1.02	**1.36**	12.92	1.02	**1.36**	16.47	1.02
ED-11-43	2153	4	2153	5.31	3.3	6.81	20.41	1.29	9.88	1.02	1.28	12.88	1.02	**1.27**	16.36	1.02
ED-11-48	2194	4	2194	9.41	3.64	11.46	20.86	1.15	12.08	1.02	1.14	16.55	1.02	**1.12**	21.72	1.02
ED-11-52	2242	4	2239	11.3	3.47	8.55	21.67	0.63	10.24	1.01	0.63	13.36	1.01	**0.62**	16.50	1.01
ED-11-73	2258	4	2257	6.46	3.47	11.74	22.06	0.30	9.34	1.00	0.30	11.36	1.00	**0.29**	13.49	1.00
ED-11-45	2265	4	2264	0.43	3.33	11.62	22.50	**0.07**	7.25	1.00	**0.07**	7.39	1.00	**0.07**	7.25	1.00
ED-11-70	2276	4	2274	3.28	3.61	8.45	22.40	0.50	10.06	1.01	0.50	12.16	1.01	**0.49**	15.48	1.01
ED-11-59	2281	4	2280	8.57	3.58	8.72	23.10	0.70	10.28	1.01	0.70	12.97	1.01	**0.68**	16.75	1.01
ED-11-77	2317	4	2317	0.10	2.88	28.42	25.64	0.09	8.13	1.00	**0.08**	8.72	1.00	**0.08**	10.22	1.00
ED-11-53	2321	4	2320	0.84	3.7	10.39	24.0	0.29	9.89	1.01	**0.28**	12.28	1.01	**0.28**	14.63	1.01
ED-11-69	2338	4	2338	4.6	3.68	7.33	25.81	0.53	11.55	1.01	0.53	10.69	1.01	**0.52**	13.77	1.01
ED-11-55	2353	4	2350	2.09	3.69	8.47	23.82	**0.59**	9.02	1.01	**0.59**	10.63	1.01	**0.59**	11.57	1.01
ED-11-75	2391	4	2390	30.02	3.86	7.41	25.1	**0.76**	9.66	1.01	**0.76**	10.80	1.01	**0.76**	12.64	1.01
ED-11-44	2434	4	2430	3.56	3.95	9.42	25.66	0.56	9.83	1.01	0.56	11.52	1.01	**0.55**	14.72	1.01
ED-11-64	2446	4	2444	13.19	3.89	11.04	26.19	0.60	10.52	1.01	0.60	12.72	1.01	**0.59**	15.59	1.01
ED-11-72	2447	4	2446	4.12	4.03	10.5	27.50	0.83	11.29	1.01	**0.82**	14.75	1.01	**0.82**	18.32	1.01
ED-11-63	2510	4	2509	17.3	4.22	12.47	27.08	0.67	12.39	1.01	0.66	15.83	1.01	**0.65**	20.58	1.01
ED-11-54	2512	4	2511	5.52	4.28	11.57	27.32	0.67	12.85	1.01	**0.66**	16.77	1.01	**0.66**	21.93	1.01
ED-11-57	2559	4	2559	19.18	4.42	11.46	28.71	0.78	12.72	1.01	**0.76**	16.95	1.01	**0.76**	22.46	1.01
ED-11-76	2581	4	2581	3.92	3.81	21.74	30.16	0.11	11.17	1.00	0.11	14.18	1.00	**0.10**	16.30	1.00
ED-11-42	2598	4	2598	2.16	4.10	27.23	30.32	0.16	11.66	1.00	0.16	32.91	1.00	**0.15**	17.24	1.00
ED-11-56	2632	4	2630	3.83	4.60	12.94	30.21	0.56	12.94	1.01	**0.55**	15.88	1.01	**0.55**	19.63	1.01
ED-11-61	2726	4	2726	6.91	4.88	7.2	32.41	0.97	14.31	1.01	0.97	17.88	1.01	**0.96**	22.94	1.02
ED-11-47	2819	4	2819	4.34	4.72	24.91	34.68	0.24	12.28	1.00	**0.23**	14.20	1.00	**0.23**	16.66	1.01
Average				13.46	1.73	13.08	$\geq$ 18.32	1.14	5.79	1.02	1.13	7.45	1.02	1.11	10.14	1.02

% The instance names have been shortened. The original names include three zeros before the first number and six zeros before the second number.

by LPKwikSort and ACP were very close, differing by no more than three percentage points. To round out the results, DeterministicKwikSort produced neither high-quality solutions nor low run-times. While KwikSort had quick run-times, they were similar to those of BestInput and Spearman's footrule, which yielded better solutions.

All things considered, BestInput had a good performance on the Biomedical and SOC data sets, but performed poorly on the TOC data set, especially when the number of input rankings was high. LPKwikSort had an excellent performance on strict rankings, but its run-time increases very fast with n, which makes it unattractive for large scale problems. Additionally, this method is only able to handle strict rankings, and its non-strict variant, LPKwikSort$_h$, does not allow the aggregate ranking to include ties, thereby limiting its general application. ACP has a very good performance on the Biomedical and SOC data sets and a dominant performance on the TOC data set. Overall, it had a very robust performance in terms of solution quality and run-times on both strict and non-strict rankings instances. Quite remarkably, none of the tested instances of up to 2,820 items exceeded 90 s in run-time, which includes the time to construct $\mathbf{X}^f$ and $\mathbf{X}^{ACP}$ and to solve the corresponding Kemeny-Agg subproblems for all ACP subsets whose size is under the threshold h. In fact, the time to calculate the CR matrix, to construct $\mathbf{X}^f$ and $\mathbf{X}^{ACP}$, and to calculate the respective approximation factor took less than 1 s for each instance of the Biomedical and SOC data sets and less than 12 s for each instance of the TOC data set. As a final note, it is important to highlight that although the approximation factors achieved by ACP are instance-specific, they are considerably lower for all 113 tested instances than the guarantees offered by any existing constant-factor approximation algorithm for Kemeny-Agg. Indeed, the worst ACP approximation factor obtained was 1.3.

6. Conclusion and future research

This paper explores the partitioning of the Kemeny aggregation problem based on Condorcet extensions. These partitioning schemes offer theoretical guarantees that enable the decomposition of certain large instances of this NP-hard problem into a set of smaller subproblems that can be solved independently. Since there may exist more than one partition that satisfies the criteria of the Condorcet extensions, we formalize the concept of the finest-Condorcet partition, which is designed to provide the highest possible computational advantages

Table 3
Performance metrics of the selected algorithms for solving instances of the SOC data set with more than 100 items; times are reported in seconds (s).

Instance	n	m	$(X_i^f)^{\max}$	KwikSort		Deterministic KwikSort		LPKwikSort		BestInput		Spearman's footrule		Approximate Condorcet Partitioning								
														$h = 30$			$h = 40$			$h = 50$		
				Gap %	Time (s)	Gap %	Time (s)	Gap %	Time (s)	Gap %	Time (s)	Gap %	Time (s)	Gap %	Time (s)	AF	Gap %	Time (s)	AF	Gap %	Time (s)	AF
ED-15-12	100	4	99	13.38	0.06	12.71	0.45	**0.42**	6.25	5.81	0.02	6.54	0.06	1.76	0.59	1.06	1.21	1.23	1.06	1.39	1.87	1.06
ED-15-42	100	4	100	14.44	0.06	10.44	0.44	**0.86**	6.16	14.94	0.02	6.67	0.05	1.4	0.64	1.04	0.95	1.31	1.04	1.26	1.84	1.04
ED-15-28	102	4	99	11.57	0.06	5.12	0.47	**0.15**	6.50	2.0	0.02	8.66	0.06	1.64	0.64	1.07	1.12	1.38	1.07	0.77	1.94	1.07
ED-15-36	102	4	100	16.69	0.06	7.83	0.48	**0**	6.41	0.2	0.02	7.57	0.06	0.94	0.61	1.08	1.24	1.33	1.08	0.69	2.00	1.08
ED-15-05	103	4	94	14.21	0.06	13.56	0.55	**0.11**	10.24	12.26	0.03	6.67	0.06	0.21	0.95	1.03	0.11	1.05	1.03	0.21	1.64	1.03
ED-11-03	103	5	90	6.93	0.06	14.67	0.47	**2.85**	6.56	10.5	0.03	7.63	0.06	2.93	1.11	1.05	2.91	1.05	1.05	2.93	1.53	1.05
ED-15-29	106	4	105	16.24	0.06	10.45	0.59	**0.32**	7.39	1.51	0.02	5.84	0.06	1.3	0.66	1.06	1.46	1.41	1.06	1.3	1.85	1.06
ED-15-07	110	4	106	16.14	0.06	15.25	0.59	**0.16**	7.86	2.46	0.02	6.71	0.08	1.15	1.06	1.05	0.74	1.13	1.05	0.57	2.00	1.05
ED-15-22	112	4	110	11.37	0.08	19.27	0.67	**0.05**	8.67	0.91	0.02	6.18	0.06	1.32	0.66	1.07	1.32	1.55	1.07	1.22	1.92	1.07
ED-15-18	115	4	112	12.08	0.08	28.01	0.70	**0**	9.28	1.34	0.03	6.53	0.08	1.39	0.73	1.06	0.76	1.61	1.06	0.58	2.02	1.06
ED-15-25	115	4	114	11.26	0.06	17.98	0.70	**0.08**	9.24	1.46	0.03	7.82	0.08	1.82	0.72	1.06	0.93	1.64	1.06	1.01	1.98	1.06
ED-15-09	115	4	115	16.93	0.06	18.84	0.69	**0.22**	9.44	1.29	0.03	5.27	0.08	1.5	1.19	1.06	1.55	1.35	1.06	0.93	2.3	1.06
ED-15-20	122	4	116	19.16	0.06	32.19	0.94	**0.19**	10.99	1.4	0.03	7.49	0.09	3.16	0.78	1.08	1.95	1.87	1.08	1.5	1.91	1.08
ED-15-17	127	4	124	14.25	0.08	10.46	1.00	**0.1**	12.72	0.37	0.03	6.87	0.10	2.03	0.86	1.07	1.69	2.03	1.07	1.29	2.00	1.07
ED-15-33	128	4	126	15.86	0.08	18.06	1.00	**0.7**	12.5	2.52	0.03	7.23	0.08	3.59	0.80	1.08	2.83	2.02	1.08	1.98	2.13	1.08
ED-15-40	131	4	131	17.81	0.09	18.55	1.14	**0.26**	13.52	0.97	0.03	9.09	0.09	1.83	0.78	1.06	1.66	1.30	1.06	1.66	2.83	1.06
ED-15-23	142	4	135	27.36	0.09	18.11	1.55	**0.03**	17.58	1.05	0.05	8.11	0.11	2.24	0.91	1.07	1.97	1.37	1.07	1.8	3.06	1.07
ED-15-32	153	4	153	17.9	0.13	21.91	2.09	**0.26**	21.56	1.05	0.05	8.61	0.13	2.64	0.94	1.07	2.28	1.59	1.07	1.46	3.67	1.07
ED-15-14	163	4	160	16.96	0.16	22.37	2.67	**0.04**	26.32	0.65	0.05	9.78	0.14	2.04	1.09	1.07	1.62	3.25	1.07	1.79	2.83	1.07
ED-15-01	240	4	240	3.49	0.28	11.84	12.13	**0.35**	84.28	9.17	0.13	4.44	0.33	0.9	1.67	1.03	0.84	2.77	1.03	0.84	7.77	1.03
ED-11-01	240	5	229	4.58	0.39	9.67	12.38	**2.27**	85.31	7.63	0.19	5.51	0.39	2.43	1.75	1.04	2.32	2.58	1.04	2.33	3.89	1.04
ED-15-03	242	4	242	5.12	0.30	11.15	13.03	**1.09**	95.98	9.16	0.11	5.92	0.33	3.83	1.75	1.10	3.18	2.80	1.12	3.19	4.44	1.12
ED-11-02	242	5	239	6.63	0.47	12.04	12.20	4.05	93.31	7.96	0.19	7.46	0.39	4.07	2.02	1.06	**3.86**	6.86	1.06	3.98	5.22	1.06
Average				13.51	0.10	15.67	1.26	0.63	14.66	4.2	0.04	7.07	0.10	2.01	0.93	1.06	1.67	1.72	1.06	1.51	2.35	1.06

% The instance names have been shortened. The original names include three zeros before the first number and six zeros before the second number.

among all such partitions. We specify the requirements of the finest-Condorcet partition, prove its uniqueness, and derive an algorithm for its construction. Condorcet partitioning is useful for a small portion of problem instances, as it often yields a few large subsets which may be too difficult to solve with exact methods. To overcome this issue, we propose Approximate Condorcet Partitioning (ACP), which breaks down these larger subsets based on the number of times an item is pairwise preferred over other items. The resulting partition has more subsets than the finest-Condorcet partition and is therefore easier to solve. Furthermore, we propose an efficient solution technique for strict and non-strict rankings, which is accompanied by instance-specific approximation factors. Although the approximation factors are not constant, ACP often achieves better solution guarantees than all known approximation factors, including all instances tested herein. The average approximation ratio for the strict and non-strict rankings instances tested herein was 1.06 and 1.03, respectively, whereas the best known approximation factors for strict and non-strict rankings are 4/3 and 2, respectively. Experiments on a variety of very large benchmark instances demonstrate the scalability and robustness of the proposed approximation algorithm. The conducted experiments on real-world instances showed that LPKiwkSort and the proposed solution technique via ACP had the best and second best performances in terms of solution quality on strict rankings, differing by no more than three percentage points; however, ACP was on average nearly six times faster than LPKiwkSort. On the other hand, ACP had a dominant performance on non-strict rankings, achieving near-optimal solutions on the majority of the tested instances.

All existing Condorcet partitioning schemes and ACP are only suitable for complete rankings (i.e., in which all items are evaluated by all judges). There are numerous group-decision making contexts where judges are unable to express their preferences over all items to produce a complete ranking; that is, they may rank only a smaller subset of all the items, and the sizes of the subsets may differ from one judge to another. Reasons for this include practicality, feasibility, and judiciousness (Moreno-Centeno and Escobedo, 2016). Two relevant examples include the National Science Foundation (NSF) proposal review process and corporate project selection (Escobedo et al., 2022). Additional potential advantages of using incomplete rankings are the mitigation of overranking fatigue and various other cognitive biases (Yoo et al., 2020).

It is generally not possible to apply the Condorcet Criterion and its extensions to incomplete rankings since the existing definitions effectively imply that each possible pair of items must be evaluated by at least one judge. In the cases when this condition is met, it may be possible to obtain a valid partition, but this remains to be formally proved and computationally tested. Future research will explore these directions as well as whether and how relaxing the partitioning conditions could expand the applicability of these methods to the general case.

CRediT authorship contribution statement

Sina Akbari: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft. **Adolfo R. Escobedo:** Supervision, Methodology, Formal analysis, Writing – review & editing.

Data availability

Public data sets.

Acknowledgment

The authors gratefully acknowledge funding support from the National Science Foundation, United States (Award 1850355). They are also grateful to the four anonymous referees, the area editor, and the editor-in-chief for their valuable and insightful feedback, which helped to enhance the clarity of this paper.

Appendix A. Proof of Theorem 1

Theorem 1. *The finest-Condorcet partition is unique.*

Proof. Let $\mathbf{X}^f, \mathbf{X}' \in \wp^f(\mathcal{X})$, where $\mathbf{X}^f = \{X_1^f, X_2^f, \dots, X_w^f\}$, $\mathbf{X}' = \{X_1', X_2', \dots, X_w'\}$, and $\mathbf{X}^f \neq \mathbf{X}'$. Since both $\mathbf{X}^f$ and $\mathbf{X}'$ are distinct finest-Condorcet partitions, they must have the same number of subsets, but the contents of some of their subsets must be different.

Consider X_1^f and X_1'. If $X_1^f = X_1'$, this part of the proof is trivially satisfied. Otherwise, assume that $X_1^f \neq X_1'$ and consider two cases based on the relative cardinality of the subsets.

Case 1. $|X_1^f| = |X_1'|$. There exist items i and j such that $i \in X_1^f, i \notin X_1'$ and $j \notin X_1^f, j \in X_1'$. This implies that i is pairwise preferred over j and j is pairwise preferred over i, yielding a contradiction.

Case 2. $|X_1^f| \neq |X_1'|$. Without loss of generality assume that $|X_1^f| > |X_1'|$.

Case 2.1: $X_1' \subset X_1^f$. In this case, the contents of the respective partitions are given by

$$\mathbf{X}^f = \{\overbrace{\{X_1', X_1^f \backslash X_1'\}}^{X_1^f}, \dots, X_w^f\},$$
$$\mathbf{X}' = \{X_1', \dots, \underbrace{\{X_1^f \backslash X_1', X_k' \backslash (X_1^f \backslash X_1')\}}_{X_k'}, \dots, X_w'\}.$$

Without loss of generality, assume that $X_1^f \backslash X_1' \subset X_{k'}$ (considering a subset of $X_1^f \backslash X_1'$ also works). $\mathbf{X}'$ is a finest-Condorcet partition and, therefore, all items in X_1' are pairwise preferred over all items in $X_1^f \backslash X_1'$; thus, it is possible to decompose X_1^f and obtain a finer partition, contradicting the assumption that $\mathbf{X}^f$ is a finest-Condorcet partition.

Case 2.2: $X_1' \not\subset X_1^f$. This leads to a similar contradiction as in Case 1, since there exists items i and j such that $i \in X_1^f, i \notin X_1'$ and $j \notin X_1^f, j \in X_1'$.

These cases prove that $X_1^f = X_1'$. Next, consider partitions $\mathbf{X}^f \backslash X_1^f$ and $\mathbf{X}' \backslash X_1'$ and apply the above chain of arguments to show that $X_2^f = X_2'$. Continuing in this manner gives that $X_k^f = X_k'$, for $k = 3, \dots, w$. Therefore, we can conclude that the finest-Condorcet partition is unique. □

Appendix B. Proof of Theorem 2

Theorem 2.

Proof. $\Longrightarrow$ We need to prove that if $\mathbf{X}^f$ satisfies Condition (4), then it is the finest-Condorcet partition. Note that $\mathbf{X}$ has the most subsets among all partitions in $\wp(\mathcal{X})$ if it is not possible to further decompose its subsets. This is indeed equivalent to satisfying Condition (4).

$\Longleftarrow$ We need to prove that the finest-Condorcet partition must satisfy Condition (4). We use contradiction. Assume that at least one of the subset of $\mathbf{X}$, say X_k, does not satisfy Condition (4). Then, we can further decompose X_k into $\overline{X}_k$ and $X_k \backslash \overline{X}_k$ and increase the size of $\mathbf{X}$ by 1. However, this contradicts the fact that $\mathbf{X}$ is the finest-Condorcet partition, as we can construct another valid partition that has more subsets. □

Appendix C. Proof of Proposition 1

Proposition 1. *Item i is pairwise preferred over item j if and only if $c_{ij} > 0$ and $c_{ji} < 0$.*

Proof. Recall that $s_{ij}, s_{ji}, t_{ij} \geq 0$, $t_{ij} = t_{ji}$, and $c_{ij} = s_{ij} + t_{ij} - s_{ji} \quad \forall i, j \in \mathcal{X}$.

$\Longrightarrow$ Assume that $s_{ij} > t_{ij} + s_{ji}$.

Case 1. $t_{ij} = 0$. By substituting $s_{ij} > s_{ji}$ in the expressions for c_{ij} and c_{ji} we have that $c_{ij} > 0$, $c_{ji} < 0$.

Case 2. $t_{ij} > 0$. By substituting $s_{ij} > s_{ji} + t_{ij}$ in the expressions for c_{ij} and c_{ji} we have:

$$c_{ij} = s_{ij} + t_{ij} - s_{ji} > s_{ji} + t_{ij} + t_{ij} - s_{ji} = 2t_{ij} > 0,$$
$$c_{ji} = s_{ji} + t_{ij} - s_{ij} < s_{ji} + t_{ij} - s_{ji} - t_{ij} = s_{ji} - s_{ji} < 0.$$

$\Longleftarrow$ Now, assume that $c_{ij} > 0,\ c_{ji} < 0$. Here, we have $c_{ij} = s_{ij} + t_{ij} - s_{ji} > 0$, which results in $s_{ij} > s_{ji} - t_{ij}$. Similarly, $c_{ji} = s_{ji} + t_{ij} - s_{ij} < 0$ results in $s_{ij} > s_{ji} + t_{ij}$. Since $s_{ji} + t_{ij} \geq s_{ji} - t_{ij}$, we can conclude that $s_{ij} > s_{ji} + t_{ij}$. □

Appendix D. Proof of Theorem 3

Theorem 3. *Algorithm 1 is correct.*

Proof. Assume that the initial partition $\mathbf{X}^0 = \{X_1^0, X_2^0, \ldots, X_w^0\}$ has been accordingly constructed, per Algorithm 1, and let $\widetilde{\mathbf{X}}$ be the working partition which is initially set to $\mathbf{X}^0$. Let $\overline{\Gamma}(\widetilde{X}_k) = \cup_{i \in \widetilde{X}_k} \overline{\Gamma}_i$ denote the set of items over which at least one of the items in $\widetilde{X}_k \in \widetilde{\mathbf{X}}$ is not pairwise preferred. If all items in $\overline{\Gamma}(\widetilde{X}_k)$ belong to $\widetilde{X}_k$ or to other lower-indexed subsets, $\widetilde{\mathbf{X}}$ does not violate NXCC; otherwise, there is a violation. Whenever a violation is detected, the associated subsets (see the next paragraph) are merged. The process continues until the working partition does not violate NXCC.

The validation & merging starts from subset $\widetilde{X}_1$. Generally, if $\overline{\Gamma}(\widetilde{X}_k) \setminus \cup_{t=1}^{k} \widetilde{X}_t = \emptyset$, all items in $\widetilde{X}_k$ are pairwise preferred over all items in $\widetilde{X}_t$, for $t = k+1, \ldots, |\widetilde{\mathbf{X}}|$. Therefore, subset $\widetilde{X}_k$ satisfies NXCC and remains unchanged in this case. If $\overline{\Gamma}(\widetilde{X}_k) \setminus \cup_{t=1}^{k} \widetilde{X}_t \neq \emptyset$, there is at least one item in subsets $\widetilde{X}_{k+1}, \ldots, X_{|\widetilde{\mathbf{X}}|}$ that not all items in $\widetilde{X}_k$ are pairwise preferred over, which causes a violation of NXCC. Let $\widetilde{X}_{k'}$ be the highest-indexed subset to which an item from $\overline{\Gamma}(\widetilde{X}_k) \setminus \cup_{t=1}^{k} \widetilde{X}_t$ belongs, where $k' > k$. Therefore, subsets $\widetilde{X}_k, \ldots, \widetilde{X}_{k'}$ are merged and placed into subset $\widetilde{X}_k$. Validation & merging is repeated for $\widetilde{X}_k$, which is now defined as $\cup_{t=k}^{k'} \widetilde{X}_t$, until $\overline{\Gamma}(\widetilde{X}_k) \setminus \cup_{t=1}^{k} \widetilde{X}_k = \emptyset$; this process is performed on the remaining subsets until the working partition satisfies NXCC. Hence, the output of the algorithm satisfies NXCC.

Furthermore, we prove that the output of the algorithm satisfies Condition (4). Recall that Condition (4) states that a subset cannot be further split into two subsets while satisfying NXCC. For this part of the proof, we emphasize that $\{X_1^f, X_2^f, \ldots, X_{|\mathbf{X}^f|}^f\} = \mathbf{X}^f$ refers to the partition output by the algorithm. Assume that subset $X_k^f \in \mathbf{X}^f$ has not undergone validation & merging, therefore, it remains unchanged after implementing the algorithm. If $|X_k^f| = 1$, then the subset trivially satisfies Condition (4). We use contradiction for the case when $|X_k^f| > 1$. Assume that subset X_k^f does not satisfy Condition (4), meaning it can be decomposed into $\overline{X}_k^f$ and $X_k^f \setminus \overline{X}_k^f$. Consider items $i \in \overline{X}_k^f$ and $j \in X_k^f \setminus \overline{X}_k^f$. This gives that $\gamma_i > \gamma_j$, contradicting the fact that $\gamma_i = \gamma_j$ (since all items with the same γ-values were placed in the same subset in $\mathbf{X}^0$ and hence $\widetilde{\mathbf{X}}$).

Now, assume that subset X_k^f has undergone the merging process and that a sequence of consecutive subsets $\{\widetilde{X}_k, \ldots, \widetilde{X}_{k'}\} \subseteq \widetilde{\mathbf{X}}$ were merged to form subset X_k^f. Additionally, assume that X_k^f does not satisfy Condition (4) and, hence, it is possible to further decompose X_k^f into $\{X_k^f, \ldots, X_t^f\}$ and $\{X_{t+1}^f, \ldots, X_{k'}^f\}$ while satisfying NXCC. This contradicts the fact that X_k^f has triggered the merging process as at least one item in X_k^f is not pairwise preferred over at least one item in $X_{k'}^f$. Therefore, X_k^f satisfies Condition (4) and Algorithm 1 is correct. □

Appendix E. Proof of Theorem 4

Theorem 4. *Algorithm 1 has a time complexity of* $O(n^2)$.

Proof. Lines 1–2 of the algorithm construct the initial partition. In line 1, calculating the Γ-parameter sets has a time complexity of $O(n^2)$, and calculating the $\overline{\Gamma}$-parameter sets and γ-values has a time complexity of $O(n)$.

Lines 5–12 perform validation & merging. In this process, the number of inner and outer while loops iterations are dependent on each other. The extreme cases are:

Case 1. the initial partition has n subsets and the output partition has n subsets. In this case, the outer while loop is performed n times but the inner while loop is never performed. In this case, lines 6–8 have a constant time, therefore, validation & merging has a time complexity of $O(n)$.

Case 2. the initial partition has n subsets and the output partition has 1 subset and each time, two adjacent subsets are merged. In this case, the inner while loop is performed $n-1$ times but the outer while loop is performed only once. Lines 10–11 have a constant time complexity, and line 12 has a time complexity of $O(n)$. Therefore, validation & merging has a time complexity of $O(n^2)$.

Finally, the finest-Condorcet partition algorithm has a time complexity of $O(n^2)$. □

Appendix F. Proof of Lemma 2

Lemma 2. *Let* $\mathbf{X} = \{X_1, X_2, \ldots, X_w\}$ *be any given partition of* $\mathcal{X}$ *and* $\hat{\boldsymbol{\sigma}}$ *be a complete ranking obtained by independently solving the subsets of* $\mathbf{X}$ *(using any method of choice) and concatenating the solutions of these subproblems. The term* $d_{KS}(\hat{\boldsymbol{\sigma}}) - d_{KS}(\boldsymbol{\sigma}^*)$ *is bounded by*

$$\begin{aligned} &2\sum_{k=1}^{w-1}\sum_{k=k+1}^{w}\sum_{i\in X_k}\sum_{j\in X_{k'}} \max\big(0, (2s_{ji}+t_{ij}) - (2s_{ij}+t_{ij}),\\ &(2s_{ji}+t_{ij}) - (s_{ij}+s_{ji})\big) +\\ &\sum_{k=1}^{w}\sum_{(i,j)\in X_k} \max\big(2s_{ij}+t_{ij}, 2s_{ji}+t_{ij}, s_{ij}+s_{ji}\big)\\ &- \min\big(2s_{ij}+t_{ij}, 2s_{ji}+t_{ij}, s_{ij}+s_{ji}\big). \end{aligned} \tag{F.1}$$

Proof. Consider an item pair (i,j), where $i \in X_k$, $j \in X_{k'}$, $k < k'$ (items from different subsets); since $\hat{\sigma}_i < \hat{\sigma}_j$, the contribution of this pair in $d_{KS}(\hat{\boldsymbol{\sigma}})$ is $2s_{ji}+t_{ij}$, while the contribution of this pair in $d_{KS}(\boldsymbol{\sigma}^*)$ must be exactly one of $2s_{ji}+t_{ij}$, $2s_{ij}+t_{ij}$, or $s_{ij}+s_{ji}$. Therefore, the additional distance accrued by (i,j) in $d_{KS}(\hat{\boldsymbol{\sigma}})$ relative to $d_{KS}(\boldsymbol{\sigma}^*)$ is at most

$$\max\Big(0, (2s_{ji}+t_{ij}) - (2s_{ij}+t_{ij}), (2s_{ji}+t_{ij}) - (s_{ij}+s_{ji})\Big).$$

Consider a pair of distinct items (i,j), where $i,j \in X_k$ (items within the same subset); the additional distance accrued by (i,j) in $d_{KS}(\hat{\boldsymbol{\sigma}})$ relative to $d_{KS}(\boldsymbol{\sigma}^*)$ is at most

$$\max\Big(2s_{ij}+t_{ij}, 2s_{ji}+t_{ij}, s_{ij}+s_{ji}\Big) - \min\Big(2s_{ij}+t_{ij}, 2s_{ji}+t_{ij}, s_{ij}+s_{ji}\Big). \tag{F.2}$$

Since the exact orderings of i and j in $\boldsymbol{\sigma}^*$ and $\hat{\boldsymbol{\sigma}}$ are not yet known, Eq. (F.2) considers the worst case. Expressly, the contribution of (i,j) in $d_{KS}(\boldsymbol{\sigma}^*)$ is taken as the smallest of the three possible values of $d_{KS}(\sigma_{ij})$, whereas the contribution of this pair in $d_{KS}(\hat{\boldsymbol{\sigma}})$ is taken as the largest of the three possible values of $d_{KS}(\sigma_{ij})$. Finally, the right-hand side of Eq. (F.1) has been multiplied by 2 since d_{KS} counts each item-pair twice. □

Appendix G. Proof of Theorem 5

Theorem 5. *Assume that the input rankings are non-strict and let* $\mathbf{X}_h^{ACP} = \{X_1, \ldots, X_w\}$ *be the ACP partition obtained using threshold* h. *Let* $\hat{\boldsymbol{\sigma}}$ *be the complete ranking obtained via ACP from the following two steps: (1) solve*

subsets of at most size h to optimality, (2) tie all items in subsets of size greater than h. The term $d_{KS}(\hat{\sigma}) - d_{KS}(\sigma^*)$ *is bounded by*

$$2\sum_{k=1}^{w-1}\sum_{k=k+1}^{w}\sum_{i\in X_k}\sum_{j\in X_{k'}}\max\left(0,(2s_{ji}+t_{ij})-(2s_{ij}+t_{ij}),\right.$$
$$\left.(2s_{ji}+t_{ij})-(s_{ij}+s_{ji})\right)+ \tag{G.1}$$
$$\sum_{k=1}^{w}\sum_{(i,j)\in X_k}\left(\left(s_{ij}+s_{ji}\right)-\min\left(2s_{ij}+t_{ij},2s_{ji}+t_{ij},s_{ij}+s_{ji}\right)\right).$$

Proof. Let $\overline{\sigma}$ be an auxiliary ranking obtained from $\mathbf{X}_h^{ACP}$, whereby all items in each subset $X_k \in \mathbf{X}_h^{ACP}$ are tied $\forall k \in \{1,\dots,w\}$ (items in lower-indexed subsets remain ranked ahead of items in higher-indexed subsets). Since $\hat{\sigma}$ solves all the subsets whose size is at most h to optimality and ties all the items in subsets whose size is greater than h, we have that $d_{KS}(\hat{\sigma}) \le d_{KS}(\overline{\sigma})$. Furthermore, we show that $d_{KS}(\overline{\sigma}) - d_{KS}(\sigma^*) \le \beta$.

Consider an item pair (i,j), where $i \in X_k$, $j \in X_{k'}$, $k < k'$ (items from different subsets); since $\overline{\sigma}_i < \overline{\sigma}_j$, the contribution of this pair in $d_{KS}(\overline{\sigma})$ is $2s_{ji}+t_{ij}$, while the contribution of this pair in $d_{KS}(\sigma^*)$ must be exactly one of $2s_{ji}+t_{ij}$, $2s_{ij}+t_{ij}$, or $s_{ij}+s_{ji}$. Therefore, the additional distance accrued by (i,j) in $d_{KS}(\overline{\sigma})$ relative to $d_{KS}(\sigma^*)$ is at most

$$\max\left(0,(2s_{ji}+t_{ij})-(2s_{ij}+t_{ij}),(2s_{ji}+t_{ij})-(s_{ij}+s_{ji})\right).$$

Consider a pair of distinct items (i,j), where $i,j \in X_k$ (items within the same subset); the additional distance accrued by (i,j) in $\overline{\sigma}$ relative to $d_{KS}(\sigma^*)$ is at most

$$\left(s_{ij}+s_{ji}\right)-\min\left(2s_{ij}+t_{ij},2s_{ji}+t_{ij},s_{ij}+s_{ji}\right). \tag{G.2}$$

Since the exact orderings of i and j in σ^* is not yet known, Eq. (G.2) considers the worst case. Expressly, the contribution of (i,j) in $d_{KS}(\sigma^*)$ is taken as the smallest of the three possible values of $d_{KS}(\sigma_{ij})$, whereas the contribution of this pair in $d_{KS}(\hat{\sigma})$ is $\left(s_{ij}+s_{ji}\right)$ as $\overline{\sigma}$ ties all items within each subset. Finally, the first term of Eq. (G.1) has been multiplied by 2 since d_{KS} counts each item-pair twice. □

Appendix H. Proof of Theorem 6

Theorem 6. *Assume that the input rankings are strict and let* $\mathbf{X}_h^{ACP} = \{X_1,\dots,X_w\}$ *be the ACP partition obtained using threshold h. Let* $\hat{\sigma}$ *be the complete ranking obtained via ACP from the following two steps: (1) solve subsets of at most size h to optimality, (2) randomly permute items within subsets of size greater than h. The term* $d_{KS}(\hat{\sigma}) - d_{KS}(\sigma^*)$ *is bounded by*

$$4\sum_{k=1}^{w-1}\sum_{k=k+1}^{w}\sum_{i\in X_k}\sum_{j\in X_{k'}}\max\left(0,s_{ji}-s_{ij}\right)$$
$$+\,2\sum_{k=1}^{w}\sum_{(i,j)\in X_k}\left(s_{ji}\mathbb{1}_{\overline{\sigma}_i<\overline{\sigma}_j}+s_{ij}\mathbb{1}_{\overline{\sigma}_j<\overline{\sigma}_i}-\min\left(s_{ij},s_{ji}\right)\right); \tag{H.1}$$

where $\overline{\sigma}$ *is an auxiliary ranking obtained by randomly permuting all items in subset* X_k, $\forall k \in \{1,\dots,w\}$ *(while ranking items in the lower-indexed subsets strictly ahead of items in the higher-indexed subsets), and* $\mathbb{1}$ *is an indicator function.*

Proof. Order the items within subset $X_k \in \mathbf{X}_h^{ACP}$ whose size is greater than h in $\hat{\sigma}$ the same as in $\overline{\sigma}$. Since $\hat{\sigma}$ solves all subsets whose size is at most h to optimality whereas $\hat{\sigma}$ randomly permutes these items, we have that $d_{KS}(\hat{\sigma}) \le d_{KS}(\overline{\sigma})$. Next, we show that $d_{KS}(\overline{\sigma}) - d_{KS}(\sigma^*) \le \beta$.

Consider an item pair (i,j), where $i \in X_k$, $j \in X_{k'}$, $k < k'$ (items from different subsets); since $\overline{\sigma}_i < \overline{\sigma}_j$, the contribution of this pair in $d_{KS}(\overline{\sigma})$ is $2s_{ji}$, while the contribution of this pair in $d_{KS}(\sigma^*)$ must be exactly one of $2s_{ji}$ or $2s_{ij}$ (when the input rankings are strict, $d_{KS}(\sigma_{ij})$ has only two possible values since the aggregate ranking is assumed to be strict as well). Therefore, the additional distance accrued by (i,j) in $d_{KS}(\overline{\sigma})$ relative to $d_{KS}(\sigma^*)$ is at most

$$\max\left(0,(2s_{ji}-2s_{ij})\right).$$

Consider a pair of distinct items (i,j), where $i,j \in X_k$ (items within the same subset); the additional distance accrued by (i,j) in $\overline{\sigma}$ relative to $d_{KS}(\sigma^*)$ is at most

$$2s_{ji}\mathbb{1}_{\overline{\sigma}_i<\overline{\sigma}_j}+2s_{ij}\mathbb{1}_{\overline{\sigma}_j<\overline{\sigma}_i}-\min\left(2s_{ij},2s_{ji}\right). \tag{H.2}$$

Since the exact orderings of i and j in σ^* is not yet known, Eq. (H.2) considers the worst case. Expressly, the contribution of (i,j) in $d_{KS}(\sigma^*)$ is taken as the smallest of the three possible values of $d_{KS}(\sigma_{ij})$, whereas the contribution of this pair in $d_{KS}(\hat{\sigma})$ is $2s_{ji}$ if $\overline{\sigma}_i < \overline{\sigma}_j$, and $2s_{ij}$ otherwise. Finally, the first term of Eq. (H.1) has been multiplied by 2 since d_{KS} counts each item-pair twice. □

Appendix I. NonStrictKwikSort algorithm

NonStrictKwikSort is a randomized recursive algorithm that adapts KwikSort to handle non-strict rankings; its pseudocode is presented in Algorithm 3. Similar to KwikSort, it starts by randomly selecting an item as pivot. Next, it compares all remaining items to the pivot item and assigns them based on their relative orderings within the input rankings to three subsets: *Left (L)*, *Right (R)*, or *Middle (Q)*. All items in the Left subset will be ranked ahead of all items in the Middle and Right subsets; all items in the Middle subset will be tied and ranked ahead of all items in the Right subset in the output solution. The algorithm applies the same procedure recursively to the Left and Right subsets until the relative ordering of all items is determined. Next, we elaborate on how the items are assigned to the three aforementioned subsets. The algorithm assigns item j to the Left subset if pivot item i is ranked ahead of j in at least $(1-\beta)$% of the input rankings, it assigns j to the Right subset if j is ranked ahead of i in at least $(1-\beta)$% of the input lists, and it assigns j to the Middle subset otherwise. Here, $0 \le \beta \le 0.5$ is a user-specified parameter, which is set to 0.25 herein. We note that parts of the algorithm are inspired also by BucketPivot, which is the modified KwikSort for the Optimal Bucket Pivot Problem (Aledo et al., 2017c; Gionis et al., 2006).

Appendix J. Objective function values of the tested algorithms

Tables J.4–J.6 report the objective function values (i.e., the cumulative Kemeny–Snell distance of the obtained solution to the input rankings) of the tested algorithms, which connect to Tables 1–3, respectively.

Table J.4
Objective function values of the tested algorithms for solving instances of the Biomedical data set.

Instance	n	m	NonStrictKwikSort	BestInput	ACP		
					$h=30$	$h=40$	$h=50$
Prostate Cancer	218	4	51,047	40,809	44,495	44,282	44,260
Bladder Cancer	308	4	49,927	42,841	42,845	42,845	42,845
Breast Cancer	386	4	145,646	113,651	139,333	139,549	139,765
Retinoblastoma	402	4	78,812	78,273	78,211	78,211	78,211
Neuroblastoma	431	4	92,065	83,835	82,881	82,626	82,507

Table J.5
Objective function values of the tested algorithms for solving instances of the TOC data set.

Instance	n	m	NonStrictKwikSort	BestInput	ACP		
					$h = 30$	$h = 40$	$h = 50$
ED-14-02	100	5,000	4,725,000	–	4,725,000	4,725,000	4,725,000
ED-14-03	100	5,000	4,651,850	–	4,651,850	4,651,850	4,651,850
MD-03-03	102	32	15,840	24,088	15,840	15,840	15,840
MD-03-05	103	31	15,500	26,244	15,500	15,500	15,500
MD-03-06	133	38	24,700	41,920	24,700	24,700	24,700
MD-03-08	147	51	43,629	68,751	43,629	43,629	
MD-03-07	155	51	38,760	46,884	38,760	38,760	38,760
ED-10-50	170	4	33,308	27,832	24,729	24,761	24,736
ED-10-49	351	4	115,864	117,059	101,164	101,164	101,060
ED-18-01	379	723	60,854,829	–	30,454,824	30,454,824	3,0454,824
ED-18-03	477	556	46,367,191	–	23,258,021	23,258,021	2,325,8021
ED-11-12	1,210	4	1,767,972	1,668,735	1,635,020	1,634702	1,634,384
ED-11-31	1,223	4	1,770,836	1,650,797	1,543,810	1,543658	1,543,203
ED-11-09	1,272	4	1,913,851	1,693,653	1,616,193	1,615724	1,614,631
ED-11-23	1,342	4	2,046,014	2,017,064	1,874,898	1,874345	1,873,792
ED-11-21	1,347	4	2,022,786	1,930,340	1,883,935	1,883384	1,883,017
ED-11-37	1,351	4	2,079,039	1,867,027	1,842,379	1,841128	1,840,414
ED-11-25	1,356	4	2,188,135	2,013,239	1,894,977	1,893496	1,892,756
ED-11-13	1,363	4	2,063,442	2,094,091	1,996,645	1,996252	1,996,056
ED-11-29	1,368	4	2,204,187	1,822,030	1,810,126	1,809600	1,808,550
ED-11-14	1,375	4	2,372,258	1,935,788	1,925,538	1,924047	1,923,301
ED-11-30	1,386	4	1,918,957	1,942,725	1,873,251	1,872336	1,871,788
ED-11-06	1,449	4	2,466,040	2,204,747	2,154,039	2,152992	2,1521,53
ED-11-04	1,467	4	3,295,480	2,336,042	2,284,842	2,284175	2,283,729
ED-11-07	1,474	4	2,391,411	2,503,234	2,348,065	2,347604	2,347,373
ED-11-34	1,509	4	2,679,187	2,537,303	2,409,181	2,408469	2,408,232
ED-11-22	1,514	4	2,665,468	2,377,686	2,287,445	2,286323	2,285,874
ED-11-11	1,545	4	3,155,771	2,402,785	2,319,852	2,318489	2,317,807
ED-11-15	1,563	4	3,359,863	2,576,355	2,470,338	2,469612	2,469,128
ED-11-08	1,572	4	2,905,609	2,888,032	2,738,763	2,738493	2,738,222
ED-11-28	1,616	4	2,791,072	3,106,683	2,737,564	2,737293	2,737,293
ED-11-40	1,623	4	3,364,856	2,958,710	2,490,748	2,490500	2,490,500
ED-11-36	1,634	4	3,522,390	2,954,359	2,774,579	2,773759	2,773,486
ED-11-33	1,646	4	3,026703	2,912747	2,8059,48	2,804847	2,803,471
ED-11-05	1,673	4	2,696,219	3,194,942	2,607,873	2,607,355	2,607,096
ED-11-18	1,681	4	3,294,078	3,115,489	2,990,565	2,990,271	2,989392
ED-11-16	1,708	4	3,113,117	3,146,307	2,946,005	2,945,423	2,945,423
ED-11-32	1,751	4	3,576,633	3,449,842	3,323,704	3,322,723	3,322,723
ED-11-38	1,754	4	3,765,400	3,50,8307	3,375,624	3,374,632	3,373,970
ED-11-39	1,788	4	3,979,326	3,475,074	2,951,711	2,951,123	2,950,535
ED-11-68	1,826	4	5,095,568	3,852,346	3,516,812	3,516,812	3,516,812
ED-11-49	1,845	4	4,026,119	4,173,178	3,946,723	3,946,332	3,946,332
ED-11-20	1,870	4	3,553,291	3,709,222	3,503,269	3,501,889	3,500,509
ED-11-26	1,931	4	3,985,229	3,946,271	3,761,869	3,760,756	3,759,643
ED-11-35	1,936	4	4,459,450	4,043,148	3,865,440	3,864,679	3,863,537
ED-11-74	1,976	4	4,416,193	4,424,128	4,224,911	4,224,076	4,223,658
ED-11-60	1,977	4	4,300,726	4,602,648	4,183,358	4,183,358	4,183,358
ED-11-58	2,011	4	4,516,635	4,683,885	4,414,887	4,414,451	4,414,451
ED-11-62	2,014	4	4,401,234	4,758,697	4,267,979	4,267,133	4,266,710
ED-11-17	2,015	4	4,659,696	4,301,476	4,109,993	4,1087,76	4,107,964
ED-11-66	2,024	4	4,612,366	4,502,283	4,369,497	4,368,641	4,368,212
ED-11-24	2,049	4	4,560,710	4,537,540	4,352,606	4,351,748	4,351,319
ED-11-67	2,066	4	4,815,256	4,810,335	4,531,663	4,522,717	4,522,717
ED-11-27	2,092	4	5,682,906	4,987,647	4,699,969	4,699,040	4,699,040
ED-11-10	2,096	4	4,624,781	4,949,710	4,383,916	4,383,045	4,382,610
ED-11-19	2,104	4	4,829,212	4,656,600	4,425,867	4,424,993	4,424,119
ED-11-50	2,111	4	4,960,102	5,270,413	4,901,645	4,901,158	4,901,158
ED-11-51	2,112	4	5,078,828	4,989,425	4,828,691	4,827,740	4,827,264
ED-11-65	2,119	4	4,889,334	5,747,053	4,754,678	4,754,204	4,754,678
ED-11-41	2,123	4	5,441,209	5,394,043	5,070,400	5,069,898	5,069,898
ED-11-71	2,127	4	5,796,474	5,340,186	4,926,537	4,926,047	4,926,047
ED-11-46	2,133	4	5,068,560	5,333,519	4,901,268	4,900,784	4,900,784
ED-11-43	2,153	4	5,274,322	5,3494,47	5,072,985	5,072,484	5,071,983
ED-11-48	2,194	4	5,536,944	5,640,689	5,118,927	5,118,421	5,117,409
ED-11-52	2,242	4	5,879,025	5,733,766	5,315,421	5,315,421	5,314,892
ED-11-73	2,258	4	5,759,327	6,044,968	5,426,081	5,426,081	5,425,540
ED-11-45	2,265	4	5,729,952	6,368,389	5,709,413	5,709,413	5,709,413
ED-11-70	2,276	4	5,808,590	6,099,357	5,652,240	5,652,240	5,651,677
ED-11-59	2,281	4	6,069,458	6,077,844	5,629,497	5,629,497	5,628,378
ED-11-77	2,317	4	4,951,261	6,352,058	4,950,767	4,950,272	4,950,272
ED-11-53	2,321	4	5,648,155	6,183,061	5,617,349	5,616,789	5,616,789
ED-11-69	2,338	4	6,229,042	6,391,616	5,986,669	5,986,669	5,986,074
ED-11-55	2,353	4	6,243,828	6,634,030	6,152,088	6,152,088	6,1520,88

(*continued on next page*)

Table J.5 (*continued*).

Instance	n	m	NonStrictKwikSort	BestInput	ACP		
					$h = 30$	$h = 40$	$h = 50$
ED-11-75	2,391	4	8,294,209	6,851,876	6,427,661	6,427,661	6,427,661
ED-11-44	2,434	4	6,664,168	7,041,263	6,471,115	6,471,115	6,470,472
ED-11-64	2,446	4	7,262,737	7,124,784	6,454,910	6,454,910	6,454,269
ED-11-72	2,447	4	6,672,648	7,081,517	6,461,804	6,461,164	6,461,164
ED-11-63	2,510	4	7,683,897	7,367,501	6,594,526	6,593,871	6,593,216
ED-11-54	2,512	4	6,843,662	7,236,044	6,529,108	6,528,459	6,528,459
ED-11-57	2,559	4	8,130,761	7,604,083	6,875,467	6,874,102	6,874,102
ED-11-76	2,581	4	6,440,721	7,545,163	6,204,586	6,204,586	6,203,966
ED-11-42	2,598	4	6,549,882	8,157,219	6,421,654	6,421,654	6,421,013
ED-11-56	2,632	4	7,468,730	8,124,033	7,233,511	7,232,792	7,232,792
ED-11-61	2,726	4	8,287,366	8,309,846	7,826,914	7,826,914	7,826,139
ED-11-47	2,819	4	8,399,175	10,055,021	8,069,133	8,068,328	8,068,328

Table J.6
Objective function values of the tested algorithms for solving instances of the SOC data set.

Instance	n	m	KwikSort	Deterministic KwikSort	LPKwikSort	BestInput	Spearman's footrule	ACP		
								$h = 30$	$h = 40$	$h = 50$
ED-15-12	100	4	7,492	7,448	6,636	6,992	7,040	6,724	6,688	6,700
ED-15-42	100	4	9,128	8,809	8,045	9,168	8,508	8,088	8,052	8,076
ED-15-28	102	4	8,714	8,210	7,822	7,966	8,486	7,938	7,897	7,870
ED-15-36	102	4	9,426	8,711	8,078	8,094	8,690	8,154	8,178	8,134
ED-15-05	103	4	4,246	4,222	3,722	4,174	3,966	3,726	3,722	3,726
ED-11-03	103	5	10,122	10,855	9,736	10,460	10,188	9,743	9,741	9,743
ED-15-29	106	4	8,592	8,164	7,416	7,504	7,824	7,488	7,500	7,488
ED-15-07	110	4	8,864	8,796	7,644	7,820	8,144	7,720	7,688	7,676
ED-15-22	112	4	9,798	10,493	8,802	8,878	9,342	8,914	8,914	8,905
ED-15-18	115	4	10,022	11,447	8,942	9,062	9,526	9,066	9,010	8,994
ED-15-25	115	4	10,990	11,654	9,886	10,022	10,650	10,058	9,970	9,978
ED-15-09	115	4	10,556	10,729	9,048	9,144	9,504	9,163	9,168	9,112
ED-15-20	122	4	14,945	16,579	12,566	12,718	13,481	12,938	12,787	12,730
ED-15-17	127	4	13,504	13,056	11,832	11,864	12,632	12,060	12,020	11,972
ED-15-33	128	4	13,790	14,052	11,985	12,202	12,763	12,329	12,239	12,138
ED-15-40	131	4	16,484	16,588	14,028	14,128	15,264	14,248	14,224	14,224
ED-15-23	142	4	18,401	17,065	14,452	14,600	15,620	14,772	14,733	14,708
ED-15-32	153	4	18,390	19,016	15,639	15,762	16,941	16,010	15,954	15,826
ED-15-14	163	4	20,859	21,823	17,841	17,950	19,578	18,198	18,123	18,153
ED-15-01	240	4	29,824	32,230	28,919	31,461	30,098	29,077	29,060	29,060
ED-11-01	240	5	32,119	33,682	31,409	33,055	32,404	31,458	31,425	31,428
ED-15-03	242	4	63,564	67,210	61,127	66,007	64,048	62,784	62,391	6,2397
ED-11-02	242	5	68,838	72,331	67,173	69,697	69,374	67,186	67,050	67,127

Algorithm 3: NonStrictKwikSort($\mathcal{V}$)

Input: $\mathcal{V}$ (a subset of items, i.e., $\mathcal{V} \subseteq \mathcal{X}$)

```
if V = ∅ then
    return empty list
Pick pivot i ∈ V uniformly at random;
Set L ← ∅, R ← ∅, Q ← {i};
for all j ∈ V\{i} do
    if s_ij / m ≥ 1 − β then
        L ← L ∪ {j};
    else if s_ji / m ≥ 1 − β then
        R ← R ∪ {j};
    else
        Q ← Q ∪ {j};
return NonStrictKwikSort(L), Q, NonStrictKwikSort(R)
```

References

Ailon, N., 2010. Aggregation of partial rankings, p-ratings and top-m lists. Algorithmica 57 (2), 284–300.

Ailon, N., Charikar, M., Newman, A., 2008. Aggregating inconsistent information: Ranking and clustering. J. ACM 55 (5), 1–27.

Akbari, S., Escobedo, A.R., 2021. Lower bounds on kemeny rank aggregation with non-strict rankings. In: 2021 IEEE Symposium Series on Computational Intelligence. SSCI, IEEE, pp. 1–8.

Aledo, J.A., Gámez, J.A., Molina, D., 2017a. Tackling the supervised label ranking problem by bagging weak learners. Inf. Fusion 35, 38–50.

Aledo, J.A., Gámez, J.A., Molina, D., 2019. Approaching the rank aggregation problem by local search-based metaheuristics. J. Comput. Appl. Math. 354, 445–456.

Aledo, J.A., Gámez, J.A., Rosete, A., 2017b. Partial evaluation in rank aggregation problems. Comput. Oper. Res. 78, 299–304.

Aledo, J.A., Gámez, J.A., Rosete, A., 2017c. Utopia in the solution of the bucket order problem. Decis. Support Syst. 97, 69–80.

Aledo, J.A., Gámez, J.A., Rosete, A., 2021. A highly scalable algorithm for weak rankings aggregation. Inform. Sci. 570, 144–171.

Azzini, I., Munda, G., 2020. A new approach for identifying the kemeny median ranking. European J. Oper. Res. 281 (2), 388–401.

Badal, P.S., Das, A., 2018. Efficient algorithms using subiterative convergence for kemeny ranking problem. Comput. Oper. Res. 98, 198–210.

Bartholdi, J., Tovey, C.A., Trick, M.A., 1989. Voting schemes for which it can be difficult to tell who won the election. Soc. Choice Welf. 6 (2), 157–165.

Betzler, N., Bredereck, R., Niedermeier, R., 2014. Theoretical and empirical evaluation of data reduction for exact kemeny rank aggregation. Auton. Agents Multi-Agent Syst. 28 (5), 721–748.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D., 2016. Handbook of Computational Social Choice. Cambridge University Press.

Cascaro, R.J., Gerardo, B.D., Medina, R.P., 2019. Aggregating filter feature selection methods to enhance multiclass text classification. In: Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City. pp. 80–84.

Cohen-Boulakia, S., Denise, A., Hamel, S., 2011. Using medians to generate consensus rankings for biological data. In: International Conference on Scientific and Statistical Database Management. Springer, pp. 73–90.

Marquis de Condorcet, M.J.A., 1785. Essai sur l'application de l'analyse a la Probabilite des Decisions: Rendues a la Pluralite de Voix. De l'Imprimerie royale.

Conitzer, V., Davenport, A., Kalagnanam, J., 2006. Improved bounds for computing kemeny rankings. In: AAAI. pp. 620–626.

Cook, W.D., 2006. Distance-based and ad hoc consensus models in ordinal preference ranking. European J. Oper. Res. 172 (2), 369–385.

Dahiya, S., Handa, S., Singh, N., 2016. A rank aggregation algorithm for ensemble of multiple feature selection techniques in credit risk evaluation. Int. J. Adv. Res. Artif. Intell. 5 (9), 1–8.
D'Ambrosio, A., Mazzeo, G., Iorio, C., Siciliano, R., 2017. A differential evolution algorithm for finding the median ranking under the kemeny axiomatic approach. Comput. Oper. Res. 82, 126–138.
Desarkar, M.S., Sarkar, S., Mitra, P., 2016. Preference relations based unsupervised rank aggregation for metasearch. Expert Syst. Appl. 49, 86–98.
Ding, J., Han, D., Yang, Y., 2018. Iterative ranking aggregation using quality improvement of subgroup ranking. European J. Oper. Res. 268 (2), 596–612.
Dodgson, C., 1876. A method of taking votes on more than two issues. In: The Theory of Committees and Elections. Cambridge University Press.
Dong, Y., Li, Y., He, Y., Chen, X., 2021. Preference–approval structures in group decision making: Axiomatic distance and aggregation. Decis. Anal. 18 (4), 273–295.
Drotár, P., Gazda, M., Vokorokos, L., 2019. Ensemble feature selection using election methods and ranker clustering. Inform. Sci. 480, 365–380.
Dwork, C., Kumar, R., Naor, M., Sivakumar, D., 2001. Rank aggregation methods for the web. In: Proceedings of the 10th International Conference on World Wide Web. pp. 613–622.
Emond, E.J., Mason, D.W., 2002. A new rank correlation coefficient with application to the consensus ranking problem. J. Multi-Criteria Decis. Anal. 11 (1), 17–28.
Escobedo, A.R., Moreno-Centeno, E., Yasmin, R., 2022. An axiomatic distance methodology for aggregating multimodal evaluations. Inform. Sci. 590, 322–345.
Gionis, A., Mannila, H., Puolamäki, K., Ukkonen, A., 2006. Algorithms for discovering bucket orders from data. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 561–566.
Gross, O.A., 1962. Preferential arrangements. Amer. Math. Monthly 69 (1), 4–8.
Kemeny, J.G., Snell, L.J., 1962. Preference ranking: An axiomatic approach. In: Mathematical Models in the Social Sciences. Ginn New York, pp. 9–23.
Kemmer, R., Yoo, Y., Escobedo, A., Maciejewski, R., 2020. Enhancing collective estimates by aggregating cardinal and ordinal inputs. In: Proceedings of the AAAI Conference on Human Computation and Crowdsourcing. pp. 73–82.
Kenyon-Mathieu, C., Schudy, W., 2007. How to rank with few errors. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing. pp. 95–103.
Laslier, J.F., 1997. Tournament Solutions and Majority Voting, 7. Springer.
Liu, N., Xu, Z., Zeng, X.J., Ren, P., 2021. An agglomerative hierarchical clustering algorithm for linear ordinal rankings. Inform. Sci. 557, 170–193.
Luce, R.D., 2012. Individual Choice Behavior: A Theoretical Analysis. Courier Corporation.
Mandal, M., Mukhopadhyay, A., 2017. Multiobjective PSO-based rank aggregation: Application in gene ranking from microarray data. Inform. Sci. 385, 55–75.
Mao, A., Procaccia, A., Chen, Y., 2013. Better human computation through principled voting. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1142–1148.
Marbach, D., Costello, J.C., Küffner, R., Vega, N.M., Prill, R.J., Camacho, D.M., Allison, K.R., Kellis, M., Collins, J.J., Stolovitzky, G., 2012. Wisdom of crowds for robust gene network inference. Nature Methods 9 (8), 796–804.
Mattei, N., Walsh, T., 2013. Preflib: A library for preferences http://www. preflib. org. In: International Conference on Algorithmic Decision Theory. Springer, pp. 259–270.
Milosz, R., Hamel, S., 2018. Exploring the median of permutations problem. J. Discrete Algorithms 52, 92–111.
Milosz, R., Hamel, S., 2020. Space reduction constraints for the median of permutations problem. Discrete Appl. Math. 280, 201–213.
Mohammadi, M., Rezaei, J., 2020. Ensemble ranking: Aggregation of rankings produced by different multi-criteria decision-making methods. Omega 96, 102254.
Moreno-Centeno, E., Escobedo, A.R., 2016. Axiomatic aggregation of incomplete rankings. IIE Trans. 48 (6), 475–488.
Oliveira, S.E., Diniz, V., Lacerda, A., Merschmanm, L., Pappa, G.L., 2020. Is rank aggregation effective in recommender systems? An experimental analysis. ACM Trans. Intell. Syst. Technol. 11 (2), 1–26.
Onan, A., 2018. Ensemble learning based feature selection with an application to text classification. In: 2018 26th Signal Processing and Communications Applications Conference. SIU, IEEE, pp. 1–4.
Plackett, R.L., 1975. The analysis of permutations. J. R. Stat. Soc. Ser. C. Appl. Stat. 24 (2), 193–202.
Puerta, J.M., Aledo, J.A., Gámez, J.A., Laborda, J.D., 2021. Efficient and accurate structural fusion of Bayesian networks. Inf. Fusion 66, 155–169.
Şahin, A., Sevim, İ., Albey, E., Güler, M.G., 2022. A data-driven matching algorithm for ride pooling problem. Comput. Oper. Res. 140, 105666.
Schalekamp, F., Zuylen, A.v., 2009. Rank aggregation: Together we're strong. In: 2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments. ALENEX, SIAM, pp. 38–51.
Tideman, N., 2017. Collective Decisions and Voting: The Potential for Public Choice. Routledge.
Truchon, M., 1998. An Extension of the Condorcet Criterion and Kemeny Orders. Citeseer.
Van Zuylen, A., Williamson, D.P., 2007. Deterministic algorithms for rank aggregation and other ranking and clustering problems. In: International Workshop on Approximation and Online Algorithms. Springer, pp. 260–273.
Wald, R., Khoshgoftaar, T.M., Dittman, D., Awada, W., Napolitano, A., 2012. An extensive comparison of feature ranking aggregation techniques in bioinformatics. In: 2012 IEEE 13th International Conference on Information Reuse & Integration. IRI, IEEE, pp. 377–384.
Yoo, Y., Escobedo, A.R., 2021. A new binary programming formulation and social choice property for kemeny rank aggregation. Decis. Anal. 18 (4), 296–320.
Yoo, Y., Escobedo, A.R., Skolfield, J.K., 2020. A new correlation coefficient for comparing and aggregating non-strict and incomplete rankings. European J. Oper. Res. 285 (3), 1025–1041.
Young, H.P., 1977. Extending condorcet's rule. J. Econom. Theory 16 (2), 335–353.
Young, H.P., 1988. Condorcet's theory of voting. Am. Political Sci. Rev. 82 (4), 1231–1244.
Young, H.P., Levenglick, A., 1978. A consistent extension of condorcet's election principle. SIAM J. Appl. Math. 35 (2), 285–300.