European Journal of Operational Research 305 (2023) 1323–1336

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

# Reducing the time required to find the Kemeny ranking by exploiting a necessary condition for being a winner[☆]

Noelia Rico[1], Camino R. Vela, Irene Díaz[*]

*Department of Computer Science, University of Oviedo, Edificio Departamental Oeste. Módulo 1, 2.ª planta. Calle Pedro Puig Adam, s/n., Campus de Gijón, C.P. 33204, Spain*

ARTICLE INFO

*Article history:*
Received 21 June 2021
Accepted 20 July 2022
Available online 25 July 2022

*Keywords:*
Group decisions and negotiations
Combinatorial optimization
Computational social choice
Ranking aggregation
Kemeny method

ABSTRACT

The ranking aggregation problem is gaining attention in different application fields due to its connection with decision making. One of the most famous ranking aggregation methods can be traced back to Kemeny in 1959. Unfortunately, the problem of determining the result of the aggregation proposed by Kemeny, known as Kemeny ranking as it minimizes the number of pairwise discrepancies from a set of rankings given by voters, has been proved to be NP-hard, which unfortunately prevents practitioners from using this method in most real-life problems. In this work, we introduce two exact algorithms for determining the Kemeny ranking. The best of these algorithms guarantees a reasonable search time up to 14 alternatives, showing an important reduction of the execution time in comparison to other algorithms found in the literature. Moreover, a dataset of profiles of rankings is provided and a study of additional aspects of the votes that may have impact on the execution time required to determine the winning ranking is also detailed.

## 1. Introduction

The need of ranking different alternatives according to the preferences of some voters arises frequently in many fields of application such as medicine (Cao, Guo, Ao, & Zhou, 2020), recommender systems (Oliveira, Diniz, Lacerda, Merschmanm, & Pappa, 2020) and quality of life research (Goerlich & Reig, 2021). For this reason, preference aggregation has caught the attention of researchers from many different fields and in particular of social choice theorists. Within the field of social choice theory, methods of different nature have been proposed to obtain the ranking that best represents the preferences of voters (Fishburn, 1973). In particular, the interest of this paper lies in the subfield of social choice theory usually referred to as computational social choice (Brandt, Conitzer, Endriss, Lang, & Procaccia, 2016), which focuses on the computation requirements and aspects related to the efficiency of these methods.

The main difficulty is then to define methods that avoid the voting paradox. In this direction, Kemeny (1959) proposed a method that searches for the ranking that is the *closest* to all rankings given by voters. This method naturally extends Condorcet's proposal in the sense that when the Condorcet ranking exists, it is the ranking obtained by the Kemeny method. Although there exist different approaches for solving the ranking aggregation problem (Schulze, 2011; Tideman, 1987), the one proposed by Kemeny is the only one that has been proved to be *neutral*, *consistent* and *Condorcet* at the same time (Hemaspaandra, Spakowski, & Vogel, 2005; Young, 1988). These desirable properties make the Kemeny method very appealing to be used in real-life problems.

In order to determine the Kemeny ranking, it is necessary to compute the distance of all the possible rankings to the rankings provided by some voters. This is necessary because the Kemeny ranking is defined as the one that minimizes the distance to the rankings given by the voters. This fact makes the complexity of the Kemeny method greatly depend on the number of alternatives. Unfortunately, there exists no known algorithm to compute the Kemeny ranking in polynomial time for any number of alternatives, which actually prevents its use in practice in many real-life problems. In fact, the problem of finding the Kemeny ranking has been proved to be NP-hard (Bartholdi, Tovey, & Trick, 1989). Some algorithms have been proposed over the years in order to reduce

[☆] This research has been supported by Grant TIN2017-87600-P from the Spanish Government and PID2019-106263RB-I00.

[*] Corresponding author.
*E-mail address:* sirene@uniovi.es (I. Díaz).

[1] Noelia Rico is supported by the Severo Ochoa program of the Principality of Asturias with the [grant PA-20-PF-BP19-167]. https://github.com/noeliarico

https://doi.org/10.1016/j.ejor.2022.07.031

the number of rankings to explore to determine the Kemeny ranking (Azzini & Munda, 2020). Along this line of research, we propose two new exact algorithms that reduce the number of rankings to explore by using some intrinsic properties related to the domain of the Kemeny problem.

The remainder of this paper is organized as follows. In Section 2, the Kemeny method and the notation used throughout this work are introduced. Section 3 discusses the use of branch and bound algorithms for computing the Kemeny ranking. More precisely, the existing branch and bound algorithms for the computation of the Kemeny ranking already described in the literature are recalled and some new proposals are introduced. The experiments conducted to measure the performance of the proposed algorithms and the obtained results are discussed in Section 4. Additional aspects concerning the impact of the characteristics of the profile of rankings on the execution time are analyzed in Section 5. Some final remarks and future lines of research are given in Section 6.

## 2. The Kemeny ranking aggregation method

Consider a set of $n$ alternatives $\mathcal{A} = \{a_1, \cdots, a_n\}$. A binary relation $R$ on the set $\mathcal{A}$ is a subset of the Cartesian product $\mathcal{A} \times \mathcal{A}$, that is, a set of ordered pairs $(a_i, a_j) \in \mathcal{A}$. It is usually noted as $(\mathcal{A}, R)$. A linear order is a binary relation that is transitive, antisymmetric and complete ($a_iRa_j \vee a_jRa_i$). The linear order $(\mathcal{A}, \succ)$ is called a **ranking**, being $\succ$ the *better than* preference on the set of alternatives $\mathcal{A}$ (Roberts & Tesman, 2009). Obviously, if $\mathcal{A}$ contains $n$ alternatives, there are $n!$ different rankings that can be obtained from the set. Note that $a_i \succ a_j$ or $a_j \succ a_i$ for each pair of alternatives $a_i, a_j \in \mathcal{A}$ in a ranking with $i \neq j$. As an example, given the set of alternatives $\mathcal{A} = \{a_1, a_2, a_3\}$, one of the 6 different rankings that can be obtained is $a_1 \succ a_3 \succ a_2$. The alternative ranked at the first position of a ranking (most left) is considered the *best* or *most preferred* alternative and, consequently, the alternative ranked at the last position is considered the *worst* or *least preferred* alternative.

We refer as **ranking with ties** to any linear order defined considering the binary relation as transitive and complete but not antisymmetric. These are also known as *weak orders.* In this kind of rankings, for all pairs of alternatives $a_i, a_j \in \mathcal{A}$ such that $i \neq j$ the relationship between the pair is known and can be *better than* ($\succ$) or indifferent ($\sim$) such that either one of the three must happen: $a_i \succ a_j$, $a_j \succ a_i$ or $a_i \sim a_j$. As an example, given the set of alternatives $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, a ranking with ties on this set could be $a_1 \succ a_2 \sim a_4 \succ a_3$, indicating that $a_1$ is the most preferred alternative, the alternatives $a_2$ and $a_4$ are equally preferred and at the same time less preferred than $a_1$ and more preferred than $a_3$, which is the least preferred alternative.

A *profile of rankings* $\pi^n_m$ is a list of $m$ rankings (with or without ties) given by $m$ different voters over the set of $n$ alternatives. A compact representation of a profile of rankings can be given under assumption of anonymity, that is, when all the voters are assumed to be equal. In this work, anonymity is assumed, which makes possible to use a compact representation of the profile that contains $m' \leq m$ different rankings, and each $r_i \in \pi^n_m$ of these unique rankings is weighted by the number $w_i$ of voters that expressed the ranking $r_i$. Thus, it holds that $m = \sum_{i=1}^{m'} w_i$.

The profile of rankings may be further simplified by just providing the pairwise comparisons between the pairs of alternatives $a_i, a_j \in \mathcal{A}$ by means of the *outranking matrix* (Arrow & Raynaud, 1986).

**Definition 1.** The outranking matrix $\mathbf{O}$ is a matrix representing the profile such that each element $o_{ij} \in \mathbf{O}$ represents the number of times that the alternative $a_i$ is preferred over the alternative $a_j$. Therefore, for rankings (with or without ties), it holds that $o_{ij} + o_{ji} = m$ for any $i \neq j$. By definition, the diagonal elements $o_{ij}$ when $i = j$ of the outranking matrix are set to 0.

**Table 1**
Profile of rankings $\pi^4_{10}$ given by ten voters on the set of four alternatives $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ (left) and corresponding outranking matrix (right).

| Number of voters | Ranking |
|---|---|
| 1 | $a_2 \succ a_3 \succ a_1 \succ a_4$ |
| 3 | $a_1 \succ a_4 \succ a_2 \succ a_3$ |
| 4 | $a_4 \succ a_1 \succ a_2 \succ a_3$ |
| 2 | $a_1 \succ a_3 \succ a_2 \succ a_4$ |

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|
| $a_1$ | 0 | 9 | 9 | 6 |
| $a_2$ | 1 | 0 | 8 | 3 |
| $a_3$ | 1 | 2 | 0 | 3 |
| $a_4$ | 4 | 7 | 7 | 0 |

**Table 2**
Points annotated to compute the Kemeny distance between the rankings $r_i$ and $r_j$ based on the order of the alternatives $a_k$ and $a_\ell$.

| | $a_k \succ_j a_\ell$ | $a_k \sim_j a_\ell$ | $a_\ell \succ_j a_k$ |
|---|---|---|---|
| $a_k \succ_i a_\ell$ | 0 | 1 | 2 |
| $a_k \sim_i a_\ell$ | 1 | 0 | 1 |
| $a_\ell \succ_i a_k$ | 2 | 1 | 0 |

Computationally, the value of the element $o_{ij}$ when $i \neq j$ is obtained from $\pi^n_m$ by annotating 1 point every time that $a_i \succ a_j$ and 0.5 points every time that $a_i \sim a_j$. An example of a profile of rankings and its corresponding outranking matrix is shown in Table 1.

The aim of a ranking aggregation function is to summarize the information of the profile of rankings into the ranking that best represents the preferences of the voters of the profile. The resulting ranking is usually referred to as the *winning ranking*. In this work, we try to find the ranking (without ties) that results from combining the rankings (with or without ties) in a profile of rankings using the Kemeny method.

Kemeny (1959) proposed a method that selects as the winning ranking from all possible rankings on $\mathcal{A}$, the ranking $s$ that minimizes the so-called Kemeny distance $\delta(s, \pi^n_m)$ from the ranking to the profile of rankings. Intuitively, this distance represents the number of pairwise discrepancies in the relative order of every pair of alternatives in the ranking $s$ and all the rankings in the profile $\pi^n_m$. Formally, given two rankings $r_i$ and $r_j$ on the set of alternatives $\mathcal{A}$, the Kemeny distance between these two rankings is computed by annotating for each pair of alternatives $a_k, a_\ell \in \mathcal{A}$ a certain number of points $d_{r_i,r_j}(a_k, a_\ell)$ according to the order of these alternatives in the rankings, as shown in Table 2.

By adding these points for every pair of alternatives, the Kemeny distance $\delta(r_i, r_j)$ between two rankings is defined as

$$\delta(r_i, r_j) = \sum_{k=1}^{n-1} \sum_{\ell=k+1}^{n} d_{r_i,r_j}(a_k, a_\ell) \,. \tag{1}$$

**Example 1.** As an example, we illustrate the computation of the Kemeny distance between the rankings $r_1 = a_3 \succ a_4 \succ a_1 \succ a_2$ and $r_2 = a_2 \succ a_3 \sim a_4 \succ a_1$. First of all, it is necessary to check the relative position of $a_1$ with respect to the three remaining alternatives. This comparison annotates $2 + 0 + 0 = 2$ points, as $a_1$ is ranked at a worse position than $a_3$ and $a_4$ in both rankings (0 points in each comparison because the order is the same) but the order of $a_1$ and $a_2$ is the opposite in $r_1$ and $r_2$ (2 points). Afterwards, $a_2$ is compared with $a_3$ and $a_4$, which annotates $2 + 2 = 4$ points because $a_2$ is ranked at a worse position than $a_3$ and $a_4$ in $r_1$ but at better position in $r_2$ (different order, 2 points each). Lastly, $a_3$ and $a_4$ are compared, adding 1 more point as both alternatives are tied in $r_2$ but not in $r_1$. This procedure results in a final Kemeny distance of $\delta(r_1, r_2) = (2 + 0 + 0) + (2 + 2) + (1) = 7$.

The Kemeny distance of a ranking $s$ to the profile of rankings $\pi^n_m$ is defined as the sum of the Kemeny distances of $s$ to all the

**Table 3**
Distance computed using the outranking matrix from all the possible rankings obtained using the set of alternatives $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ to the profile of rankings in Table 1.

| Ranking | $\delta$ | Ranking | $\delta$ | Ranking | $\delta$ | Ranking | $\delta$ |
|---|---|---|---|---|---|---|---|
| $a_1 \succ a_4 \succ a_2 \succ a_3$ | **14** | $a_1 \succ a_3 \succ a_4 \succ a_2$ | 24 | $a_4 \succ a_3 \succ a_1 \succ a_2$ | 30 | $a_2 \succ a_3 \succ a_1 \succ a_4$ | 38 |
| $a_4 \succ a_1 \succ a_2 \succ a_3$ | 16 | $a_4 \succ a_2 \succ a_1 \succ a_3$ | 24 | $a_3 \succ a_1 \succ a_4 \succ a_2$ | 32 | $a_4 \succ a_3 \succ a_2 \succ a_1$ | 38 |
| $a_1 \succ a_2 \succ a_4 \succ a_3$ | 18 | $a_2 \succ a_1 \succ a_4 \succ a_3$ | 26 | $a_4 \succ a_2 \succ a_3 \succ a_1$ | 32 | $a_2 \succ a_3 \succ a_4 \succ a_1$ | 40 |
| $a_1 \succ a_4 \succ a_3 \succ a_2$ | 20 | $a_1 \succ a_3 \succ a_2 \succ a_4$ | 28 | $a_3 \succ a_4 \succ a_1 \succ a_2$ | 34 | $a_3 \succ a_4 \succ a_2 \succ a_1$ | 42 |
| $a_1 \succ a_2 \succ a_3 \succ a_4$ | 22 | $a_2 \succ a_4 \succ a_1 \succ a_3$ | 28 | $a_3 \succ a_1 \succ a_2 \succ a_4$ | 36 | $a_3 \succ a_2 \succ a_1 \succ a_4$ | 44 |
| $a_4 \succ a_1 \succ a_3 \succ a_2$ | 22 | $a_2 \succ a_1 \succ a_3 \succ a_4$ | 30 | $a_2 \succ a_4 \succ a_3 \succ a_1$ | 36 | $a_3 \succ a_2 \succ a_4 \succ a_1$ | 46 |

rankings $r_i \in \pi^n_m$, as shown in Eq. (2).

$$\delta(s, \pi^n_m) = \sum_{i=1}^{m'} w_i \cdot \delta(s, r_i)\,. \tag{2}$$

The Kemeny problem is actually a minimization problem, as for all the possible $n!$ rankings of the $n$ alternatives in $\mathcal{A}$, the one (or ones) that minimizes the value of $\delta$ is (or are) selected as the winning ranking. In the remainder of this paper, we skip the factor 2 in Eq. (3) as it obviously does not affect the minimization problem.

Equivalently, the Kemeny distance between a ranking $s$ and the profile of rankings may also be computed from the outranking matrix. This can be achieved by summing the elements of the matrix that show disagreement with the ranking $s$. More formally, for two alternatives $a_i, a_j$, the element $o_{ij}$ shows the agreement of the voters with the order $a_i \succ a_j$ and, conversely, the element $o_{ji}$ shows the disagreement of the voters with the order $a_i \succ a_j$. The Kemeny distance of the ranking $s$ to the profile of rankings $\pi^n_m$ is then given in terms of the outranking matrix $\mathbf{O}$ as follows:

$$\delta(s, \pi^n_m) = 2 \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} o_{ji} \cdot x_{ij} \quad \text{with}\, i \neq j\,, \tag{3}$$

where $x_{ij} = 1$ if $a_i \succ_s a_j$ and 0 otherwise; which represents the sum of all the elements $o_{ji}$ such that $a_j$ is ranked at a worse position than $a_i$ in the ranking $s$.

**Example 2.** Consider the ranking $r_1 = a_1 \succ a_3 \succ a_2 \succ a_4$. The distance of the ranking to the profile of rankings in Table 1 can be computed using Eq. (3) as follows.

- Firstly, the alternative $a_i, i = 1$ is considered, then $x_{1j} = 1$ for $1 < j \leq n$. Thus, the sum of all the elements in the first column of the matrix $(1 + 1 + 4 = 6)$ represents the disagreement with $a_1$ being ranked at the first position. Thus, all the rankings with $a_1$ in the top position, i.e., of the form '$a_1 \succ \ldots$' will have at least a distance to the profile of rankings of 6, obtained after adding $o_{2,1} = 1$, $o_{3,1} = 1$ and $o_{4,1} = 4$.
- Secondly, as $a_3$ is in the second position let us consider next $i = 3$, having $x_{3j} = 1$ for $j = 2$ and $j = 4$, as $a_2$ and $a_4$ are ranked in a worse position. For this reason, the sum of all the elements in the third column of the matrix except the first one (because $a_1 \succ a_3$ and therefore $o_{1,3}$ does not represent disagreement) are added $(o_{2,3} + o_{4,3} = 8 + 7 = 15)$. The elements added are the ones that represent disagreement with $a_2$ being ranked at the second position when $a_1$ is already ranked at the first position.
- Finally, the fourth element of the second column $(o_{4,2} = 7)$ represents the disagreement with $a_2$ being ranked over the alternative $a_4$. This results in a Kemeny distance (recall that we omit the factor 2, as it does not affect the optimization problem) of $\delta(r_1, \pi^n_m) = ((1 + 1 + 4) + (8 + 7) + (7)) = 28$.

The distances obtained using this simplification from all the possible rankings to the profile of rankings in Table 1 are shown in Table 3.

From Example 2 it can be seen that the rankings that share the top alternatives will use the same elements of the outranking matrix associated to these alternatives in order to partially compute the distance from the ranking to the profile. Notice how, using Eq. (3), it is possible to associate a common partial distance for all the rankings that have the same top $\ell$ alternatives in the same order.

**Definition 2.** Let $\mathcal{A} = \{a_1, \cdots, a_n\}$ be a set of alternatives and let $a_{\sigma(1)} \succ a_{\sigma(2)} \succ \cdots \succ a_{\sigma(\ell)} \succ a_{\sigma(\ell+2)} \succ \cdots \succ a_{\sigma(n)}$ be a ranking with $\sigma$ a permutation of $\{1, 2, \ldots, n\}$. Then, $\rho = a_{\sigma(1)} \succ a_{\sigma(2)} \succ \cdots \succ a_{\sigma(\ell)}$ is a *prefix* of length $\ell$.

**Definition 3.** Let $\mathcal{A} = \{a_1, \cdots, a_n\}$ be a set of alternatives. Let $\rho = a_{\sigma(1)} \succ a_{\sigma(2)} \succ \cdots \succ a_{\sigma(\ell)}$ be a prefix. The Kemeny partial distance from a prefix to a profile $\pi^n_m$ is defined as

$$\delta_\rho(\pi^n_m) = \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} o_{\sigma(j)\sigma(i)} \cdot x_{\sigma(i)\sigma(j)} \quad \text{with}\, \sigma(i) \neq \sigma(j)\,, \\ \sigma \text{ a permutation on the alternatives}$$

Given $\rho$, there are $(n - \ell)!$ rankings on $\mathcal{A}$ that start with this prefix.

**Proposition 1.** *Let $\rho$ be a prefix of length $\ell$, then*

$\delta_\rho(\pi^n_m) \leq \delta(r^\rho, \pi^n_m)$,

*for each ranking $r^\rho = \rho \succ a_{\sigma(\ell+1)} \succ \cdots \succ a_{\sigma(n)}$.*

**Proof.** It is straightforward as all the elements of the outranking matrix are non negative. Thus, by Def. 3 and Eq. (3)

$$\delta_\rho(\pi^n_m) = \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} o_{\sigma(j)\sigma(i)} \cdot x_{\sigma(i)\sigma(j)} \leq \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} o_{\sigma(j)\sigma(i)} \cdot x_{\sigma(i)\sigma(j)} \\ + \sum_{i=\ell+1}^{n}\sum_{j=\ell+1}^{n} o_{\sigma(j)\sigma(i)} \cdot x_{\sigma(i)\sigma(j)} = \delta(r^\rho, \pi^n_m)$$

□

Using Example 2 again to illustrate Proposition 1, it is shown that, if $\rho := a_1 \succ a_2$, using the profile of rankings $\pi^4_{10}$ given in Table 1, then the distance from this prefix to the profile is obtained from adding $o_{2,1} + o_{3,1} + o_{4,1}$ and then $o_{3,2} + o_{4,2}$ obtaining $\delta_\rho = 1 + 1 + 4 + 2 + 7 = 15$. Further example of this can be observed in Table 3, as the rankings that have the same first and second alternative only differ in their distance taking into account the third alternative $a_i$ and $a_j$, so they differ only in the value $o_{ji}$.

The ranking that minimizes the Kemeny distance to the profile of rankings is typically referred to as the Kemeny ranking. This ranking has also been referred to as the maximum likelihood ranking by some authors (Young, 1988), as the Kemeny problem can be reformulated as the problem of seeking for the ranking that maximizes the agreement with the profile of rankings.

Although along this paper we refer to *the Kemeny ranking* in singular, it is necessary to keep in mind that the problem may have more than one solution depending on the profile of rankings.

## 3. Algorithms to solve the Kemeny problem

A naive approach to solve the Kemeny problem would require to explore all the $n!$ possible rankings over the set of alternatives $\mathcal{A}$ in order to determine the one that is the closest to the profile of rankings. This exploration of all the possible solutions of the Kemeny problem has a main drawback that prevents practitioners from using this method in real-life problems, as the computation of the solution, both in terms of execution time and memory storage, becomes unavoidably. Note that in this work we consider that the solution to the Kemeny problem is a strict ranking, therefore not containing tied alternatives, which is a common practice in the literature. Obviously, the number of possible rankings with tied alternatives is even greater than $n!$ (see, e.g., Bailey, 1998), therefore the consideration of rankings with tied alternatives as possible solutions would lead to an even less-attractive problem from a computational point of view.

The process of searching the Kemeny ranking can be framed as a *combinatorial optimization problem*, as it is based on finding an optimal solution from a finite set of tentative solutions. Within this framework, the set of all tentative $n!$ solutions of the problem is referred to as the *search space* of the problem and denoted by $S$. As already mentioned, this exhaustive search is not tractable, therefore it is necessary to carefully reduce the search space and avoid the exploration of some of the possible rankings.

A possible way out for solving the problem is to consider *branch-and-bound (BB)*. This kind of algorithms aims to optimize the value of an *objective function* by recursively splitting the search space into different branches, which are subsequently pruned when it is proved that they cannot lead to an optimal solution. In the context of the Kemeny problem, in order to determine the winning ranking for a profile of rankings $\pi^n_m$ over the set of alternatives $\mathcal{A}$, the search space $S$ is defined as the set of all possible rankings $s$ on $\mathcal{A}$ and the objective function to optimize is the Kemeny distance $\delta(s, \pi^n_m)$. The aim is to minimize the value of this function in order to find the Kemeny ranking.

Different branch-and-bound algorithms have been proposed to find the Kemeny ranking. Emond & Mason (2002) considered also weak orders as potential solutions and proposed to create branches for each pair of alternatives defining the three possible pairwise relations (better, worse, tied) for the pair of alternatives, subsequently associating each branch with a correlation measure and penalty. Amodio, D'Ambrosio, & Siciliano (2016) further explored this idea. Barthelemy, Guenoche, & Hudry (1989) introduced the exploration of the rankings with the same *prefix*, i.e. the branches group the rankings that rank the first alternatives in the same order. These prefixes have also been used by Muravyov (2007, 2013). In a recent work, Azzini & Munda (2020) provided a necessary condition for alternatives to be at the top of the winning ranking and proposed a recursive algorithm based on this condition. Although they did not frame their algorithm as a BB algorithm, it can be understood as one, as it is based on pruning the search space according to the aforementioned necessary condition for an alternative to be ranked at the top position. Azzini and Munda's algorithm will be explained in detail in the upcoming sections since it will be used as base for the algorithms proposed in this work.

### 3.1. BB algorithms based on prefixes

Branch and bound algorithms that build the rankings by adding one alternative in each step are a strong option for solving the Kemeny problem. As introduced in Definition 2, a prefix of length $\ell$ represents the order for the alternatives in the top $\ell$ positions. Given a profile of rankings on a set of $n$ alternatives, the search space is modeled as a rooted tree structure of $n$ levels where each level $\ell \in \{1, \dots, n\}$ of the tree contains nodes representing all different prefixes of length $\ell$. Therefore, each node has an associated *prefix* $\rho$ that represents the subset of the search space containing all the possible rankings beginning with that prefix[2] (Muravyov, 2013). Each prefix has an associated partial distance $\delta_\rho$ to the profile of rankings, which defines the *lower bound* for that branch, i.e., all the rankings beginning with that prefix are at least at a distance $\delta_\rho$ from the profile of rankings.

Using this structure, the first level of the tree contains $n$ nodes, where each node has a prefix of length one containing one different alternative $a_i \in \mathcal{A}$. Each node of the tree at the $\ell$-th level has $n - \ell$ successors, that are obtained by adding at the end of the prefix $\rho$ one of the alternatives in $\mathcal{A}$ that had not been added yet to the prefix $\rho$. The leaves of the tree at the $(\ell = n)$-th level are rankings, therefore their associated $\delta_\rho$ represents the distance of the corresponding possible solution to the profile of rankings. If the tree is not pruned, there are $n!$ leaves and all of them should be explored in order to determine the solution of the problem. These solutions are the leaf nodes that minimize $\delta_\rho$.

Branch and bound algorithms ensure that no optimal solution is lost by using lower and upper bounds. Each node has an associated lower bound and, in case that this lower bound exceeds the upper bound, then the branch can be safely pruned as it cannot lead to an optimal solutions.

The nodes are explored in a depth-first fashion, i.e., all the children of the node $\rho$ that can lead to a potential solution are explored before the siblings of $\rho$. The aim of the branch and bound algorithm is to prune the branches of this tree using information of the problem in order to reduce the number of explored nodes.

Let $\delta_\rho = \delta_\rho(\pi^n_m)$ be the partial distance from the prefix $\rho$ to the profile of rankings (see Proposition 1) and let us denote by $\delta^*$ the best distance for a ranking solution $s \in S$ found until the moment during the exploration of the search space, i.e. the lowest value of $\delta_\rho$ found in a leaf node. The value $\delta^*$ is the *upper bound* used to prune the tree, as the solutions that exceed this bound are not optimal solutions because at least one solution with lower distance to the profile has been found. As it was detailed in Section 2, the distance $\delta_\rho$ associated with a prefix cannot decrease when a new alternative is appended to the prefix. Therefore, only the nodes whose associated $\delta_\rho$ does not exceed the upper bound $\delta^*$ (i.e., $\delta_\rho \leq \delta^*$) need to be explored, as they are the only ones that can lead to a solution that improves the current one.

By using this process, $\delta^*$ is modified at the beginning of the algorithm with the distance of the first solution $s$ explored and subsequently updated every time that a leaf node that minimizes the distance to the profile is reached. At the end of the execution, the value of $\delta^*$ is the minimum distance to the profile of rankings.

The steps of this branch and bound algorithm using $\delta$ to prune the search space are presented in Algorithm 1.

Notice that the default initialization of $\delta^*$ can be represented as $\delta^* = +\infty$, which makes $\delta^*$ to take the value of the first leaf node explored as previously explained.

### 3.2. Necessary condition for the winning alternative

Azzini & Munda (2020) proved in Proposition 1 of their work that a necessary condition for a ranking to be the Kemeny ranking is that the alternative $a_i \in \mathcal{A}$ at the top position must satisfy that $\sum_{j=1}^{n} o_{ij} \geq \sum_{j=1}^{n} o_{ji}$. Let us emphasize that this property is a necessary but not sufficient condition, meaning that not all the alternatives that have a row sum greater than or equal to their corresponding column sum are top alternatives.

For the sake of simplicity, in the remaining of this paper we denote by $\alpha_i$ the truth value obtained from the Boolean expression

[2] Henceforth, $\rho$ will be used interchangeably for denoting the prefix and the node associated with the prefix.

**Algorithm 1:** BB algorithm based on prefixes with prune based on $\delta$.

```
input: Initial distance $\delta^*$. By default, $\delta^* = +\infty$.
  1. Define the empty list of solutions $\tau$.
  2. Initialize the stack of nodes to explore with $n$ nodes,
     each of them contains a prefix $\rho$ of
     length $\ell = 1$ with a different alternative $a_i \in \mathcal{A}$.
  3. Take (and remove) the first node $\rho$ from the stack and
     compute the distance $\delta_\rho := \delta_\rho(\pi_m^n)$.
       • If $\ell = n$ and $\delta_\rho = \delta^*$, then add this solution to $\tau$.
       • If $\ell = n$ and $\delta_\rho < \delta^*$, then overwrite $\tau$ with the current
         ranking and update $\delta^* := \delta_\rho$.
       • If $\ell < n$ and $\delta_\rho \leq \delta^*$, then add at the beginning of the
         stack all the possible children of
         this node, obtained by appending each of the $n - \ell$
         alternatives that are not already fixed
         in $\rho$ at the end of the prefix.
  4. Repeat recursively Step 3 until the stack is empty.
```

$$\alpha_i = \sum_{j=1}^{n} o_{ij} \geq \sum_{j=1}^{n} o_{ji} \ , \tag{4}$$

which means that $\alpha_i = \text{True}$ when the alternative $a_i$ is preferred over the remaining alternatives in $\mathcal{A}$ at least as many times as the other alternatives are preferred over $a_i$.

Based on this condition, Azzini and Munda define the MorkExact algorithm to solve the Kemeny problem by recursively exploring the alternatives verifying that $\alpha_i = \text{True}$.[3] By using this algorithm, they obtain a tractable execution time up to $n = 13$ alternatives. To achieve these results, they perform experiments on profiles of rankings synthetically generated, reaching profiles with 13 alternatives and with a constant number of 100 voters.

Although they just refer to their algorithm as a *recursive exact algorithm*, this is in fact a branch and bound algorithm that has been implemented recursively. In this paper, we characterize this algorithm as such, in order to focus on how their condition allows to prune the search space and can be combined with additional pruning criteria. Furthermore, in Rico, Vela, Pérez-Fernández, & Díaz (2021b) a slightly modification of the original definition of Azzini and Munda's algorithm in order to ensure that all the solutions of the problem are taken into account. In comparison to the implementation presented in Rico et al. (2021b), the one presented right after has been further improved in order to reduce the execution time. As the sum of all the values at the $i$-th row and all the values at the $i$-th column is constant for any $i$, the value of $\alpha_i$ may be equivalently defined in terms of a threshold, as follows:

$$\alpha_i = \sum_{j=1}^{n} o_{ij} \geq \frac{m \cdot (n-1)}{2}. \tag{5}$$

Notice that the value on the right part of the comparison is constant for a fixed dimension of the matrix and number of voters. In this work, these values for all the levels of the recursive calls are computed in advance. So this computation is done only once at the beginning of the execution and each recursive call only needs to check the row sum of its associated matrix, avoiding the computation of the column sum.

The resulting algorithm after adding these considerations is referred to as ME (after the original name MorkExact) and outlined in Algorithm 2. This algorithm is considered as benchmark in this work.

**Algorithm 2:** ME.

```
input: An outranking matrix $\mathbf{O}$ for $n > 3$ alternatives.
  1. Define an empty list of tentative solutions $\tau$.
  2. Take $\mathbf{O}$ and define the set $\eta$ containing the alternatives
     in $\mathbf{O}$ with $\alpha_i = \text{True}$.
  3. Initialize a stack of nodes to explore. Create one node for
     each alternative $a_i \in \eta$.
  4. Take (and remove) the first node $\rho$ from the stack.
  5. Consider the submatrix $\mathbf{O}_\rho$, which contains only the
     alternatives of $\mathcal{A}$ that are not already fixed in $\rho$.
       • If the dimension of the matrix is $2 \times 2$. For the two
         remaining alternatives $a_i, a_j$:
           – If $o_{ij} > o_{ji}$, add to $\tau$ the ranking $\rho \succ a_i \succ a_j$.
           – If $o_{ji} > o_{ij}$, add to $\tau$ the ranking $\rho \succ a_j \succ a_i$.
           – If $o_{ij} = o_{ji}$, add to $\tau$ both rankings $\rho \succ a_i \succ a_j$
             and $\rho \succ a_j \succ a_i$.
       • If the number of alternatives in $\mathbf{O}$ is $n \geq 3$:
           – Determine $\eta$ for the subset matrix $\mathbf{O}_\rho$ containing
             the alternatives in $\mathbf{O}_\rho$ with $\alpha_i = \text{True}$ and add
             one node of the form '$\rho \succ a_i$' to the stack for each
             $a_i \in \eta$.
  6. Repeat Step 4 and Step 5 until the stack is empty.
  7. Compute $\delta(s_i, \pi_m^n)$ for all the rankings $s_i \in \tau$.
  8. Establish $\delta_{min} = \min_{s_i \in \tau}(\delta(s_i, \pi_m^n))$.
  9. Delete all rankings with value $\delta(s_i, \pi_m^n)$ greater than the
     minimum value $\delta_{min}$.
```

### 3.3. ME-RCW algorithm

In Rico et al. (2021b) a preliminary version of a new algorithm that avoids the exploration of some tentative solutions recursively checking for the existence of a *Condorcet winner* is introduced. A Condorcet winner is an alternative such that it is preferred over all the other alternatives in the profile by a simple majority of voters. As in the rankings in the profile of rankings all voters define the relationship for each pair of alternatives in $\mathcal{A}$, this means that the Condorcet winner is the alternative that is preferred over all other alternatives in the profile by half of votes. It is possible that the Condorcet winner do not exist and, on the other hand, if it exists the profile of rankings may have a Condorcet winner even if it does not have a Condorcet ranking. As a Condorcet method, the Kemeny method ensures that the Kemeny ranking ranks the Condorcet winner at the first position of the winning ranking whenever it exists. Although it is not possible for a profile of rankings to have more than one Condorcet winner, it is possible to have more than one alternative $a_i$ with $\alpha_i = \text{True}$. In this case, in presence of a Condorcet winner it is not necessary to explore all the alternatives with $\alpha_i = \text{True}$ as only the ranking starting with the Condorcet winner can lead to an optimal solution. The existence of a Condorcet ranking is guaranteed as follows.

**Definition 4.** Let $\mathbf{O}$ be an outranking matrix associated to $\pi_m^n$. The B-outranking matrix, $\mathbf{O}^B$, is a boolean matrix such that $o_{ij}^B = 1$ and $o_{ji}^B = 0$ if and only if $o_{ij} > o_{ji}$.

[3] In their original work, the authors approach this as a maximization problem that searches for the ranking with a maximum score based on the maximum likelihood interpretation of the problem. We have reformulated their interpretation as a minimization problem in order to be consistent with the other algorithms that are discussed within this paper.

**Proposition 2.** *Let $\pi_m^n$ be a profile of rankings and $\mathbf{O}^B$ the B-outranking matrix associated to* $\mathbf{O}$. $\mathbf{O}^B$ *is transitive if $o_{ij} > o_{ji}$ and $o_{jk} > o_{kj}$ then $o_{ik} > o_{ki}$ for all $i, j, k < n$.*

**Proof.** if $o_{ij} > o_{ji}$ and $o_{jk} > o_{kj}$ and thus, $o_{ik} > o_{ki}$ then it is satisfied that if $o^B_{ij} = 1$, $o^B_{jk} = 1$ then $o^B_{ik} = 1$ for all $i, j, k < n$. As consequence $\mathbf{O}^B$ is transitive. □

It is straightforward that if $\mathbf{O}^B$ represents a complete, transitive and asymmetric relation with $\binom{n}{2}$ elements equal to 1, the relation it represents is a Codorcet ranking. Example 3 illustrates how the Condorcet ranking can easily be obtained from a complete, transitive and asymmetric $\mathbf{O}^B$ by avoiding the recursive exploration process. Note also that in case that $o_{ij} = o_{ji}$ for some $i, j$, then $\mathbf{O}^B$ does not represent a complete relation and thus it is not possible to reach a Condorcet ranking, even if the relation is transitive.

**Example 3.** Let us consider the outranking matrix associated with the profile of rankings shown in Table 1. In Fig. I, the first table shows the outranking matrix and the right table the B-outranking matrix. Each alternative $a_i$ obtains an associated score by adding all the elements in the $i$-th row. The Condorcet ranking is obtained by sorting decreasingly the values. This process guarantees the profile to be a Condorcet ranking, as all the alternatives are preferred over all the other alternatives ranked in a worse position.

Moreover, from the illustrated above, it can be seen that using this matrix it can also be determined the existence of a Condorcet winner if any.

**Algorithm 3:** ME-RCW.

```
input: An outranking matrix O for $n > 3$ alternatives.
 1. Define an empty list of tentative solutions $\tau$.
 2. Take O and define the set $\eta$ containing the alternatives
    in O with $\alpha_i$ = True.
 3. Initialize a stack of nodes to explore. Create one node for
    each alternative $a_i \in \eta$.
 4. Take (and remove) the first node $\rho$ from the stack.
 5. Consider the submatrix $\mathbf{O}_\rho$, which contains only the
    alternatives of $\mathcal{A}$ that are not already fixed in $\rho$.
      • If the dimension of the matrix is $2 \times 2$. For the two
        remaining alternatives $a_i, a_j$:
          – If $o_{ij} > o_{ji}$, add to $\tau$ the ranking $\rho \succ a_i \succ a_j$.
          – If $o_{ji} > o_{ij}$, add to $\tau$ the ranking $\rho \succ a_j \succ a_i$.
          – If $o_{ij} = o_{ji}$, add to $\tau$ both rankings $\rho \succ a_i \succ a_j$
            and $\rho \succ a_j \succ a_i$.
      • If the number of alternatives in O is $n \geq 3$:
          – If there is a Condorcet winner $a_i$, add a node for
            the prefix $\rho \succ a_i$ at the beginning of the stack.
          – If there is not a Condorcet winner, determine $\eta$ for
            the subset matrix $\mathbf{O}_\rho$ containing the alternatives in
            $\mathbf{O}_\rho$ with $\alpha_i$ = True and add one node of the form
            '$\rho \succ a_i$' to the stack for each $a_i \in \eta$.
 6. Repeat Step 4 and Step 5 until the stack is empty.
 7. Compute $\delta(s_i, \pi_m^n)$ for all the rankings $s_i \in \tau$.
 8. Establish $\delta_{min} = \min_{s_i \in \tau}(\delta(s_i, \pi_m^n))$.
 9. Delete all rankings with value $\delta(s_i, \pi_m^n)$ greater than the
    minimum value $\delta_{min}$.
```

|  | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|
| $a_1$ | 0 | 9 | 9 | 6 |
| $a_2$ | 1 | 0 | 8 | 3 |
| $a_3$ | 1 | 2 | 0 | 3 |
| $a_4$ | 4 | 7 | 7 | 0 |

|  | $a_1$ | $a_2$ | $a_3$ | $a_4$ |  |
|---|---|---|---|---|---|
| $a_1$ | 0 | 1 | 1 | 1 | $\Longrightarrow 3$ |
| $a_2$ | 0 | 0 | 1 | 0 | $\Longrightarrow 1$ |
| $a_3$ | 0 | 0 | 0 | 0 | $\Longrightarrow 0$ |
| $a_4$ | 0 | 1 | 1 | 0 | $\Longrightarrow 2$ |

$a_1 \succ a_4 \succ a_2 \succ a_3$

**Fig. I.** B-outranking matrix and Condorcet ranking obtained from the profiles of rankings in Table 1.

**Proposition 3.** *In presence of a Condorcet winner, the B-outranking matrix has exactly one alternative with a total score of $n - 1$.*

**Proof.** Note that if there is a Condorcet winner, an alternative is preferred over all the others and thus $\exists i_1 / \sum_{j=1}^{n} o^B_{i_1 j} = n - 1$. Assume there are two different rows, $i_1$, $i_2$, such that

$$\sum_{j=1}^{n} o^B_{i_1 j} = n - 1, \quad \sum_{j=1}^{n} o^B_{i_2 j} = n - 1.$$

Thus,

$$\sum_{j=1}^{n} (o^B_{i_1 j} + o^B_{j i_1}) > n - 1,$$

which is a contradiction by construction of $\mathcal{O}^B$. □

Given a prefix, the ranking that minimizes the total distance to the profile of rankings must be the ranking created with the subset of alternatives not included in the prefix that minimizes the distance to the profile. Consequently, in presence of a Condorcet winner in this subset, this alternative must be the first ranked alternative in the ranking to append to the prefix and there is no need to explore the remaining alternatives in the branch. The resulting algorithm, presented in Algorithm 3, is hereinafter referred to as ME-RCW (after the original name MorkExact and the incorporation of checking if there is a Recursive Condorcet Winner). Its main characteristics are:

- In presence of a Condorcet ranking, the winning ranking can be determined by using the outranking matrix by counting how many times each alternative $a_i \in \mathcal{A}$ is preferred over other alternatives (because in this case the outranking matrix must be transitive). This property is used as a precondition in the exploration process, as this exploration must be avoided in presence of a Condorcet ranking. This means that, Kemeny method always returns Condorcet ranking if this exists, therefore, the exploration of the rankings is not necessary and the Kemeny ranking can be computed using the outranking matrix in polynomial time.
- As a Condorcet method, the Kemeny ranking is such that, if there exists a Condorcet winner for the profile of rankings, then this alternative will be ranked at the first position. Consequently, the exploration of all other alternatives is no longer needed and it is skipped, even if they fulfill that $\alpha_i$ = True.

As the Algorithm ME, the Algorithm ME-RCW may be divided also in two different parts. Firstly, all the tentative solutions $\tau$ that could lead to the Kemeny ranking are computed. Once this reduced list of tentative solutions has been obtained, the distance of each of these rankings $s \in \tau$ to the profile of rankings is computed. Considering these distances, it is necessary to filter $\tau$ in order to keep only the rankings that are the closest to the profile.

Note that the last three steps in Algorithm 2 and Algorithm 3 are needed since the property concerning the alternatives such $\alpha_i = \text{True}$ proved by Azzini & Munda (2020) is necessary but not sufficient for an alternative to be ranked at the first position. Therefore, after the recursive process, the solution cannot be determined until the distance for all the rankings in the list of tentative solutions has been computed.

The authors of the original Mork-Exact algorithm themselves realized that these final steps are among the main sources of computational inefficiency of the algorithm. This of course implies also inefficiency during the recursive process, as unnecessary nodes have been evaluated.

A more thorough study of the behavior of the algorithm for different types of profiles of rankings has been conducted in this work in comparison to the study presented in Rico et al. (2021b).

### *3.4. Proposed algorithms combining different pruning techniques*

The aim of this work is to combine the pruning techniques of these algorithms in order to obtain a reduction of the execution time.

The first step is to incorporate the necessary condition for the top alternative of the Kemeny ranking into the classic BB approach that prunes the search space based on the upper bound $\delta^*$ presented in Algorithm 1. This allows to skip the branches that cannot lead to a solution better than one found during the previous exploration of the search space. Also, solutions that would require to append alternatives with $\alpha_i = \text{False}$ to the prefix are also discarded. This version of the algorithm is referred to as ME-BB after MorkExact (the original name) adding Branch and Bound based on the distance, and it is presented in Algorithm 4.

Notice how with this algorithm, the last three steps of Algorithm ME-RCW are no longer needed since at the end of the execution $\tau$ contains only the list of final solutions.

An additional consideration that may be added to ME-BB is the one already incorporated to ME-RCW that, in the presence of the Condorcet winner among the remaining alternatives not fixed in $\rho$, only this alternative is explored. This incorporation results in the algorithm hereinafter referred to as ME-BBRCW as it incorporates branch and bound based on the distance and also the recursive Condorcet winner to the MorkExact algorithm, which is detailed in Algorithm 5.

**Algorithm 4:** ME-BB.

```
1. Define an empty list of solutions $\tau$.
2. Take $\mathbf{O}$ and define the set $\eta$ containing the alternatives
   in $\mathbf{O}$ with $\alpha_i = \text{True}$.
3. Initialize a stack of nodes to explore. Create one node for
   each alternative $a_i \in \eta$.
4. Take (and remove) the first node $\rho$ from the stack.
5. Compute the distance $\delta_\rho := \delta_\rho(\pi^n_m)$.
   • If $\delta_\rho > \delta^*$ go to Step 4.
   • If $\delta_\rho \leq \delta^*$ go to Step 6.
6. Consider the submatrix $\mathbf{O}_\rho$, which contains only the
   alternatives of $\mathcal{A}$ that are not
   already fixed in the prefix $\rho$:
   • If the dimension of the matrix is $2 \times 2$.
     – For the two remaining alternatives $a_i, a_j$:
       ∗ If $o_{ij} > o_{ji}$, define $r_1 := \rho \succ a_i \succ a_j$.
       ∗ If $o_{ji} > o_{ij}$, define $r_1 := \rho \succ a_j \succ a_i$.
       ∗ If $o_{ij} = o_{ji}$, define $r_1 := \rho \succ a_i \succ a_j$ and
         $r_2 := \rho \succ a_j \succ a_i$.
     – Compute the distance $\delta_{r_1} := \delta(r_1, \pi^n_m)$:
       ∗ If $\delta_{r_1} < \delta^*$, then empty $\tau$ and add $r_1$ to it. If $r_2$
         is defined add it also to $\tau$.
         Update $\delta^* = \delta_{r_1}$.
       ∗ If $\delta_{r_1} = \delta^*$, then append this solution to $\tau$. If $\delta_{r_2}$
         is defined add it also $\tau$.
   • If the number of alternatives in $\mathbf{O}$ is $n \geq 3$, determine
     $\eta$ for the subset matrix $\mathbf{O}_\rho$
     containing the alternatives in $\mathbf{O}_\rho$ with $\alpha_i = \text{True}$ and
     add one node of the form
     '$\rho \succ a_i$' to the stack for each $a_i \in \eta$.
7. Repeat steps 4 to 6 until the stack is empty.
```

**Example 4.** As an example of the reduction of the search space performed by the algorithm ME-BBRCW, consider the profile of rankings shown in Table 4. Note that this profiles holds that $a_1 \succ a_2$, $a_2 \succ a_3$ and $a_3 \succ a_1$. Therefore, as the outranking matrix associated with the profile is not transitive, this profile does not yield a Condorcet ranking. There is no associated Condorcet winner either, as all the alternatives lose at least once against one of the other alternatives in a pairwise context. Therefore, the algorithm must explore the search space to determine the ranking obtained with the alternatives in $\mathcal{A}$ that minimizes the distance to the profile. Fig. 1 illustrates the reduction of the search space according to the proposed algorithms.

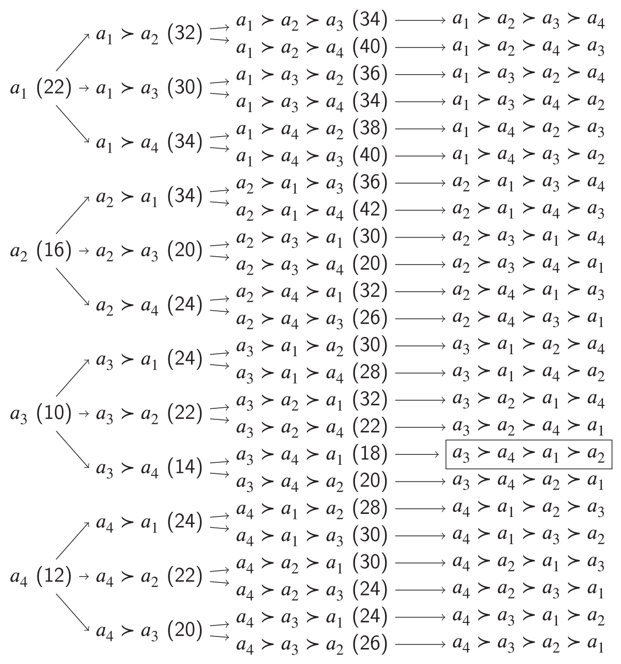
First of all, in Fig. 1(a) the complete search space that a naive algorithm would explore to find the Kemeny solution for the profile of rankings given in Table 4 is shown.

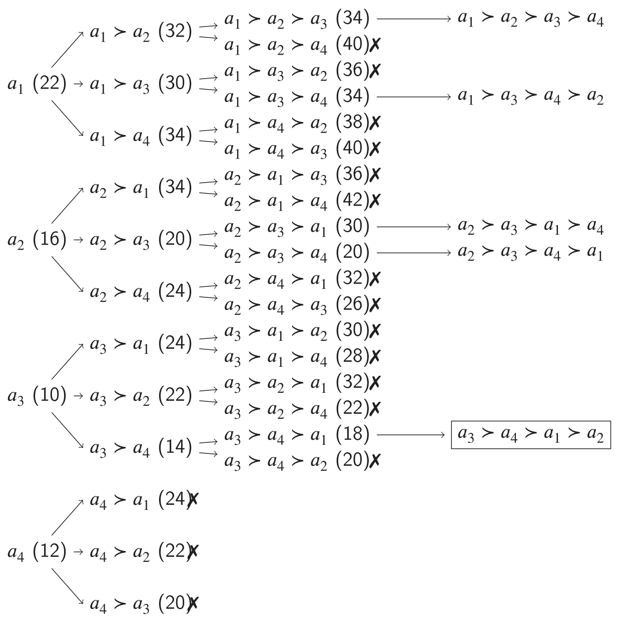
The search space when the branches are pruned according to the value of $\delta$ is shown in Fig. 1(b). In this subfigure, the algorithm initializes with a generic value of $\delta^* = +\infty$. The first ranking explored modifies this value to $\delta^* = 34$. Considering this upper bound two branches are pruned, and, subsequently, a new ranking with $\delta^* = 34$ is added to $\tau$. When a new ranking with $\delta_\rho = 30$ is found, as $\delta_\rho < \delta^*$, the list of solutions is emptied and, next, this solution is added and the value of $\delta^* = 30$ is updated. The process continues until the optimal solution with $\delta^* = 18$ is found. For profiles of rankings for which there exist more than one solution to the Kemeny problem, only those solutions would have been explored by the algorithm after this point.

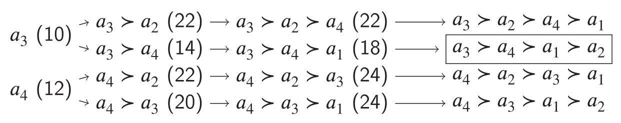
In Fig. 1(c) the search space is defined by exploring only the nodes with the alternatives that fulfill the property $\alpha_i = \text{True}$, as proposed by Azzini & Munda (2020). In this example, after finding the optimal solutions, two additional solutions are still added even if they do not improve the best found solution. At the end of the exploration, the list $\tau$ contains the four rankings. Thus, the Kemeny distance must be computed in order to find the optimal value and thus identify the rankings that are closest to the profile.

Fig. 1 (d) shows the total reduction of the search space achieved by the algorithm proposed in this paper. Notice that, when fixed the alternative $a_3$, the alternative $a_4$ is a Condorcet winner so it is the only alternative explored even if other one satisfies that $\alpha_i = \text{True}$. In this case, as the profile of rankings results in a unique Kemeny ranking, no other complete ranking is explored as the branches are pruned by using the upper bound $\delta^*$. If the Kemeny problem admits more than one solution, only those solutions are added to the list $\tau$.

The reduction shown in Example 4 is even more accentuated when both the depth of the tree and the number of recursive calls increase, causing the number of nodes in the initial search space to increase and therefore emphasizing the effect of the pruning.


(a) Complete search space showing all the possible rankings that can be obtained. The numbers in between brackets represent the partial distance $\delta_\rho$ associated with each node.


(b) Search space pruned by using the distance bound. The children of the nodes such that $\delta_\rho > \delta^*$ are no longer considered, as they may not lead to an optimal solution.


(c) Prune using ME. Only the alternatives verifying that $\alpha_i$ = True are added to $\rho$ as children, as they are the only ones that may possibly lead to an optimal solution.

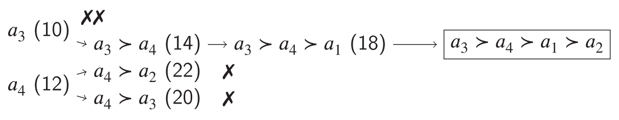

(d) Total reduction of ME-BBRCW. Branches are pruned if they try to concatenate an alternative such that $\alpha_i \neq$ True. Nodes such that $\delta_\rho > \delta^*$ are not expanded. In the presence of a Condorcet winner, only this alternative is considered.

**Fig. 1.** Search space defined in order to determine the Kemeny ranking for the profile of rankings in Table 4 using different algorithms.

## 3.5. Complexity of the algorithms

The complexity of algorithms is usually studied using the so-called Big O (Arora & Barak, 2009) notation, which defines how the algorithm responds, in terms of theoretical efficiency, when the size of the input changes.

Note that the computation time of the algorithms proposed in this work depends on a precondition (the existence of a Condorcet ranking). Thus, the best scenario happens if there is a Condorcet ranking. In this case the winning ranking is computed using the B-outranking matrix (Definition 4) in constant time. However, when there is not a Condorcet ranking, the algorithm must build the tree for searching the optimal solution according to the Kemeny method. Then, the complexity of this process must be studied.

The search space explored using BB algorithms heavily depends on the prune criteria specifically defined for the problem and on how it can be applied to a concrete input data. As consequence, the size of the elements to explore cannot be determined beforehand which makes it impossible to exactly define the complexity. Instead, the complexity of the best and worst cases are usually provided as lower and upper bounds.

Regarding the worst case, note that BB algorithms are generally applied to NP-hard problems (as in this work), for which the complexity associated to the worst case is equal to a brute force exploration. This means that the worst scenario actually gives no insight into the real performance of the algorithm. According to this, the worst complexity for the algorithms proposed in this work is $O(n!)$, as the number of rankings is factorial on the number of alternatives.

Regarding the best case, let us try to further study the complexity by modeling the search space like the tree graph that is defined by the algorithms. The theoretical complexity time can be given in terms of $k$, which corresponds with the remaining number of alternatives left to add to the prefix being currently explored, and therefore it varies at each level of the tree. Consider $t$ the time required to access one element of the outranking matrix. Then, the time complexity can be expressed using the following formulas.

$$T(k) = \begin{cases} t & \text{when } k = 1 \\ kt + kT(k-1) & \text{when } 1 < k < n \\ kT(k-1) & \text{when } k = n \end{cases}$$

To illustrate this behavior, an example with a set of $n = 4$ alternatives $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ is shown in Fig. 2. In the root node

**Algorithm 5:** ME-BBRCW.

```
input: An outranking matrix $\mathbf{O}$ for $n > 3$ alternatives.
1. Define an empty list of solutions $\tau$.
2. Take $\mathbf{O}$ and define the set $\eta$ containing the alternatives
   in $\mathbf{O}$ with $\alpha_i$ = True.
3. Initialize a stack of nodes to explore. Create one node for
   each alternative $a_i \in \eta$.
4. Take (and remove) the first node $\rho$ from the stack.
5. Compute the distance $\delta_\rho := \delta_\rho(\pi_m^n)$.
   • If $\delta_\rho > \delta^*$ go to Step 4.
   • If $\delta_\rho \leq \delta^*$ go to Step 6.
6. Consider the submatrix $\mathbf{O}_\rho$, which contains only the
   alternatives of $\mathcal{A}$ that are not already fixed in $\rho$:
   • If the dimension of the matrix is $2 \times 2$.
     – For the two remaining alternatives $a_i, a_j$:
       ∗ If $o_{ij} > o_{ji}$, define $r_1 := \rho \succ a_i \succ a_j$.
       ∗ If $o_{ji} > o_{ij}$, define $r_1 := \rho \succ a_j \succ a_i$.
       ∗ If $o_{ij} = o_{ji}$, define $r_1 := \rho \succ a_i \succ a_j$ and
         $r_2 := \rho \succ a_j \succ a_i$.
     – Compute the distance $\delta_{r_1} := \delta(r_1, \pi_m^n)$:
       ∗ If $\delta_{r_1} < \delta^*$, then empty $\tau$ and add $r_1$ to it. If $r_2$
         is defined add it also to $\tau$. Update $\delta^* = \delta_{r_1}$.
       ∗ If $\delta_{r_1} = \delta^*$, then append this solution to $\tau$. If $\delta_{r_2}$
         is defined add it also $\tau$.
   • If the number of alternatives in $\mathbf{O}$ is $n \geq 3$:
     – If there is a Condorcet winner $a_i$, add a node for
       the prefix '$\rho \succ a_i$' at the beginning of the stack.
     – If there is not a Condorcet winner, determine $\eta$ for
       the subset matrix $\mathbf{O}_\rho$ containing the alternatives in
       $\mathbf{O}_\rho$ with $\alpha_i$ = True and add one node of the form
       '$\rho \succ a_i$' to the stack for each $a_i \in \eta$.
7. Repeat steps 4 to 6 until the stack is empty.
```

**Table 4**
Profile of rankings $\pi^4_{10}$ given by ten voters on the set of four alternatives $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ (left) and corresponding outranking matrix (right).

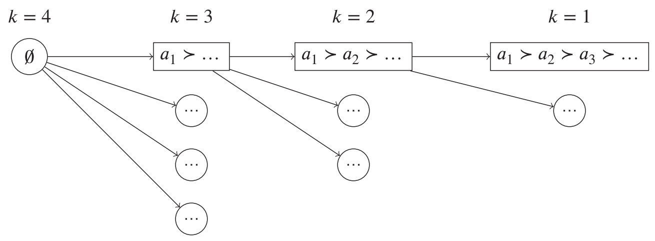
| Number of voters | Ranking |
|---|---|
| 4 | $a_2 \succ a_3 \succ a_4 \succ a_1$ |
| 4 | $a_3 \succ a_4 \succ a_1 \succ a_2$ |
| 2 | $a_4 \succ a_1 \succ a_2 \succ a_3$ |

|  | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|
| $a_1$ | 0 | 6 | 2 | 0 |
| $a_2$ | 4 | 0 | 6 | 4 |
| $a_3$ | 8 | 4 | 0 | 8 |
| $a_4$ | 10 | 6 | 2 | 0 |

of the tree, four alternatives are left to explore. Therefore $k = n = 4$. This node is an special case because neither a ranking nor a distance is associated to it. Then, a prefix $\rho$ is fixed using the first alternative of the set. When this alternative is fixed, using Definition 3 the three elements of the matrix $o_{2,1}, o_{3,1}, o_{4,1}$ must be added to the distance associated to the prefix. The same process is followed when new alternatives are added at other levels of the tree. Note that this should be repeated on the $k$ alternatives in each level of the tree, obtaining the complexity.

The results of the time complexity are shown in terms of $t$ for each level of the tree below:

$$T(1) = t$$
$$T(2) = 2t + 2T(1) = 2t + 2t = 4t$$
$$T(3) = 3t + 3T(2) = 3t + 3(4t) = 3t + 12t = 15t$$
$$T(4) = 4t + 4T(3) = 4t + 3(15t) = 49t$$



$k = 4$
$k = 3$
$k = 2$
$k = 1$
$\emptyset$
$a_1 \succ \dots$
$a_1 \succ a_2 \succ \dots$
$a_1 \succ a_2 \succ a_3 \succ \dots$

**Fig. 2.** Exploration path for the first ranking in lexicographical order of the search space and showing the number of branches expanded by each node depending on its level.

However, let us recall that the BB algorithms prune the search space. During this process, the number of branches is reduced in each node by a factor $f_k$, being $f_k$ the number of branches that can be discarded in each node of the search space with $0 \leq f_k \leq k$.

$$T(k) = \begin{cases} t & \text{when } k = 1 \\ kt + (k - f_k)T(k-1) & \text{when } 1 < k < n \\ (k - f_k)T(k-1) & \text{when } k = n \end{cases}$$

Notice that, because of how the tree is created, there is at least one node in each level which $f_k < k$, which guarantees that at least one leaf node is reached. The best case complexity in this situation happens when the two following conditions occur at the same time: 1) the first leaf node explored of each level has $f_k = k - 1$ and the remaining ones have $f_k = 0$, and 2) the ranking reached has the minimum distance to the profile and moreover it is the only solution. This leads to explore the first leaf, which corresponds with the lexicographical order of the set of the alternatives. The best case scenario is found when this ranking is the optimal and unique solution. In addition, its associated distance must be lower than the distance associated to all the nodes with $k = n - 1$. Furthermore, this distance is smaller than the distance associated to all the nodes with the same prefix of any length. In this case the complexity is linear on the number of alternatives, which represents the lower bound.

Some other considerations can be highlighted with regard to the complexity. In fact, although it is impossible to determine the number of branches that will be pruned from the search space using these algorithms, some properties of the problem can be used in order to bound it. Consider the *complementary ranking*, that is the one that has exactly the opposite relation for each pair of alternatives (given $r_1 := a_2 \succ a_3 \succ a_1 \succ a_4$, its complementary ranking is $\overline{r_1} := a_4 \succ a_1 \succ a_3 \succ a_2$). Taking into account Eq. (3), the elements of the outranking matrix considered for determining the distance $\delta(r_1, \pi)$ are exactly the opposite to the elements used for determining the distance $\delta(\overline{r_1}, \pi)$. This makes constant the sum of the distances of two complementary rankings:

$$\delta(r_1, \pi) + \delta(\overline{r_1}, \pi) = \frac{n^2 - n}{2} \cdot m. \tag{6}$$

Thus, only those alternatives $a_i$ which $\alpha_i$ = True are explored, which always leads to the complementary ranking with minimum distance. In addition, it can be guaranteed that only half of the rankings are explored. Although this does not affect the theoretical complexity as the exploration of $\frac{n!}{2}$ rankings instead of $n!$ rankings still has a complexity $O(n!)$, this has an impact on the execution time.

Another consideration is that, given the outranking matrix, the branches of the search space pruned in the first level of the tree that define the search space when $k = n - 1$ can be known prior to execution, as it is equal to the number of rows that have $\alpha$ = True (see Eq. (5)), and thus $f_{k=n-1}$ is known. However, for deeper branches, the reduction is also improved but the value of $f_k$ cannot be known in advance. Moreover, more branches will be discarded also considering the upper bound defined according to the distances previously obtained. The complexity of the operations required for branching and compute the bounds of the algorithm is, in the worst case, quadratic. Moreover notice how the number of alternatives decreases when the tree is deeper, meaning that the execution time also decreases even if the operations are still quadratic.

## 4. Experiments and results

In order to test the proposed algorithms, a synthetic dataset containing different profiles of rankings has been created. It contains only profiles of rankings that do not have a Condorcet winner (and consequently, neither a Condorcet ranking). These profiles of rankings consider numbers of alternatives $n \in [8, 14]$ and numbers of voters $m \in \{10, 50, 100, 250, 500, 1000, 2000\}$. The same numbers of voters plus one has also been considered in order to compare the behavior for an even or odd number of voters. For each different combination of $(n, m)$, a total of 200 profiles are considered. Each of the profiles has been created according to the following steps (1) Randomly select the number $d \le m$ of different rankings in the profile. (2) Obtain $d$ random different permutations of the $n$ alternatives. (3) A random vector of $d$ elements whose sum is equal to $m$ is generated where each element represents the number of voters associated to each ranking generated in the second step. (4) If the profile do not have a Condorcet winner, create the outranking matrix and add this to the dataset.

The experiments have been conducted in two separate parts. Firstly, the proposed algorithms have been compared with the original algorithm for values of $n \in \{8, 9, 10\}$ in order to check the reduction in the execution time. Secondly, the execution time for the profiles of rankings with greater values of $n$ are calculated by using the best of the proposed algorithms.

All the results presented in this paper have been obtained using a MacBookAir10,1 with a chip Apple M1 of 8 cores (with 4 performance cores and 4 efficiency cores) and 16 GB of RAM memory.

### *4.1. Comparison with the original method*

The execution times obtained for the algorithms ME-RCW, ME-BB and ME-BBRCW are compared with that for the original algorithm ME. A thorough study has been carried out for the profiles of the aforementioned synthetic dataset corresponding to the number of alternatives $n \in \{8, 9, 10\}$. This study has been restricted to these lower numbers of alternatives as the execution time is very fast, whereas for greater numbers of alternatives the execution time becomes too slow for the benchmark algorithm ME. For each profile of rankings and algorithm, the execution time required to determine the Kemeny ranking has been measured three times and the median value has been obtained with the aim of minimizing the impact that other processes being executed on the computer may have.

A general comparison of the average time required by each algorithm for each number of alternatives is shown in Fig. 3, showing the improvement of the algorithms proposed in this paper in comparison to the original algorithm. The left-hand side of the figure shows the execution times on a shared scale for all the numbers of alternatives. Note that this figure evidences the exponential growth of the execution time of the algorithms as the number of alternatives increases. The right-hand side of the figure shows the percentage of time required by the proposed algorithms to solve the Kemeny problem on average in comparison to the execution time for the Algorithm ME. Although it reduces the execution time, Algorithm ME-RCW is the worst of the three algorithms proposed in this paper, as in the worst-case scenario it requires on average a 33% of the execution time taken by the Algorithm ME. The Algorithm ME-BB requires percentages of time lower than 24% in average in comparison to the original Algorithm ME. The best improvement is achieved by ME-BBRCW, the one that combines all the pruning criteria. Its execution time is lower than the 11% of the original algorithm, independently of the number of alternatives tested. Furthermore, the improvements with respect to the Algorithm ME seem to be larger as the number of alternatives increases.

Fig. 4 shows the percentage of time required by each of the proposed algorithms in comparison to the original Algorithm ME for different numbers of alternatives and voters. Note that the behavior of Algorithm ME-RCW depends on whether the number of voters is odd or even. In addition, its execution time decreases as the number of voters increases. In contrast to Algorithm ME-RCW, the execution time for Algorithm ME-BB increases for larger numbers of voters, as recursively checking the existence of the Condorcet winner is not applied in this case. Algorithm ME-BBRCW, that combines all the pruning criteria, achieves a much more stable execution time when the number of voters varies within each number of alternatives. Nevertheless, this algorithm still presents a slight difference depending on whether the number of voters is odd or even. Furthermore, the reduction in the execution time for this algorithm is close to 10% of the execution time of the original algorithm for any combination of number of alternatives and number of voters. In addition, the improvement seems to be more substantial as the value of $n$ increases.

Paired t-tests have been performed to compare the execution time obtained by the ME-BBRCW and by the other algorithms. The p-value obtained is lower than 2.2e-16 in all the test, showing that ME-BBRCW execution time is statistically lower than the obtained by the other algorithms.

### *4.2. Results for larger numbers of alternatives*

The execution time of Algorithm ME drastically increases as the number of alternatives increases, making the comparison more difficult for large datasets. In this section, results for $n \in \{11, 12, 13, 14\}$ are shown only for the Algorithm ME-BBRCW, as it has been proved to be the fastest of all the proposed algorithms, as detailed in the previous subsection.

Fig. 5 illustrates the increment of the mean execution time as $n$ increases. Note that the value of these execution times is affected by the behavior of some profiles of rankings that present a larger execution time than the others. If instead of the mean, the median execution times were considered, the results for the values of $n$ equal to 11, 12 and 13 would be 0.246, 1.13 and 5.218, respectively. This results in almost half of the values obtained when considering the mean.

The execution time for Algorithm ME-BBRCW for different numbers of alternatives and voters is illustrated in Fig. 6. These plots show the mean execution time for each pair $(n, m)$. The horizontal line shows the mean execution time for all the profiles of rankings with the same value of $n$ in the dataset. For all values of $n$, the mean execution time for profiles of rankings with an odd and even number of voters with one unit of difference is in general lower for the odd value. Notice however how this difference tends to decrease as the number of voters increases. Although the execution time varies for each $m$ within each plot, this variation becomes less relevant as the number of voters increases, barely changing the execution time for some values of $n$ for the profiles of rankings with 2001 voters in comparison with the profiles of rankings with 10 voters.

The implemented method also runs for profiles of rankings with $n = 14$ and $n = 15$ alternatives. To illustrate the computational efficiency of the method, 200 profiles of rankings with the constant

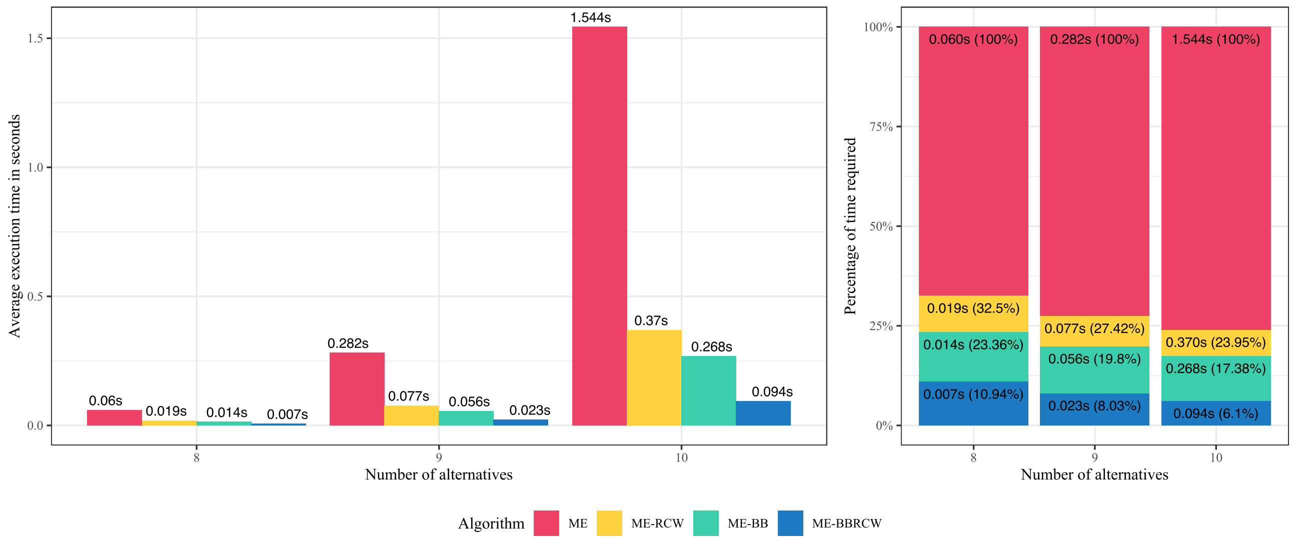


**Fig. 3.** Comparison of the average execution time for all the profiles for different numbers of alternatives (8, 9 or 10). The left-hand side of the figure illustrates the exponential increase of the execution time and the right-hand side of the figure shows the percentage of time required by the proposed algorithms in comparison to the original algorithm.

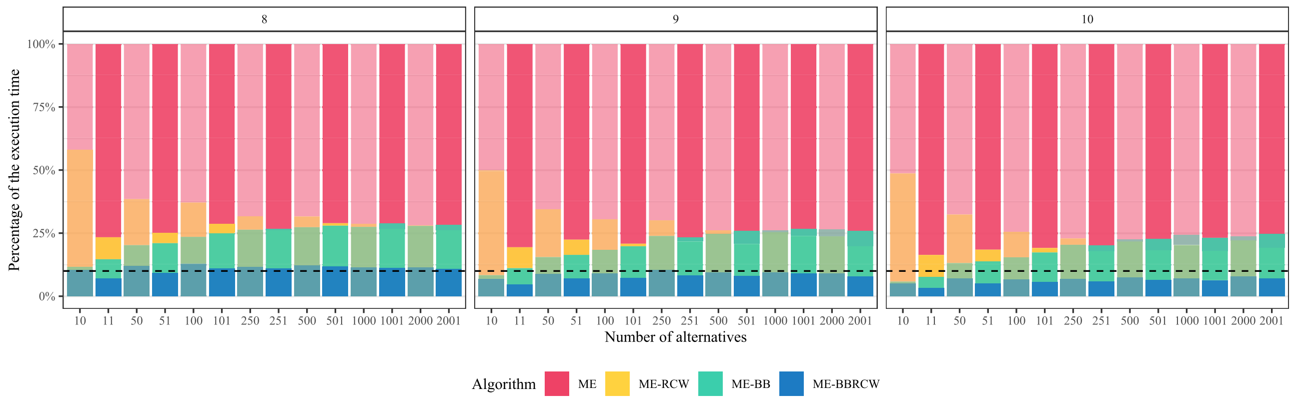


**Fig. 4.** Average percentage of time required by the algorithms for each number of alternatives and voters in comparison to the execution time of the original algorithm. The dashed horizontal line marks the 10% of the execution time in comparison to the time taken by the Algorithm ME.

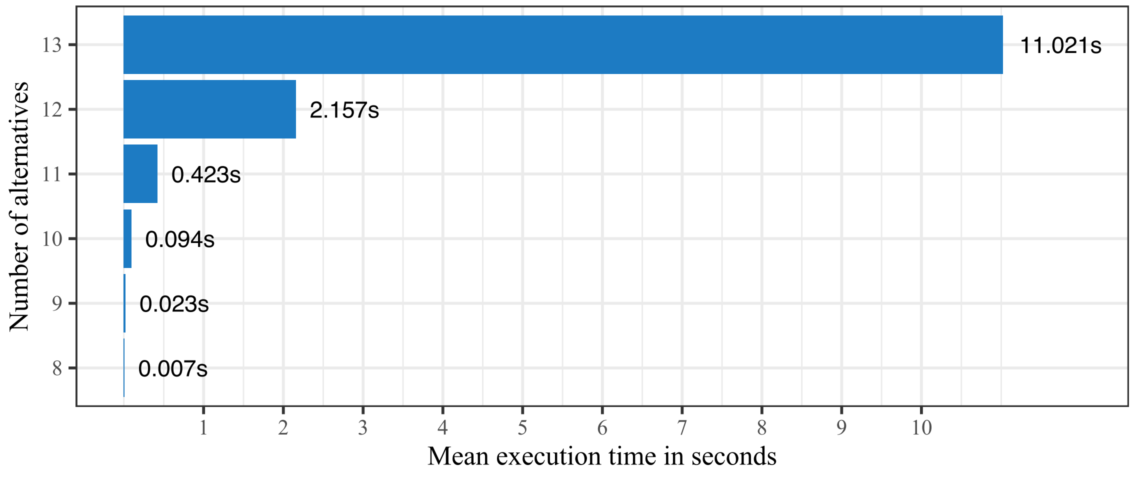


**Fig. 5.** Mean execution time in seconds for the profiles of rankings with different numbers of alternatives (averaged over all numbers of voters).

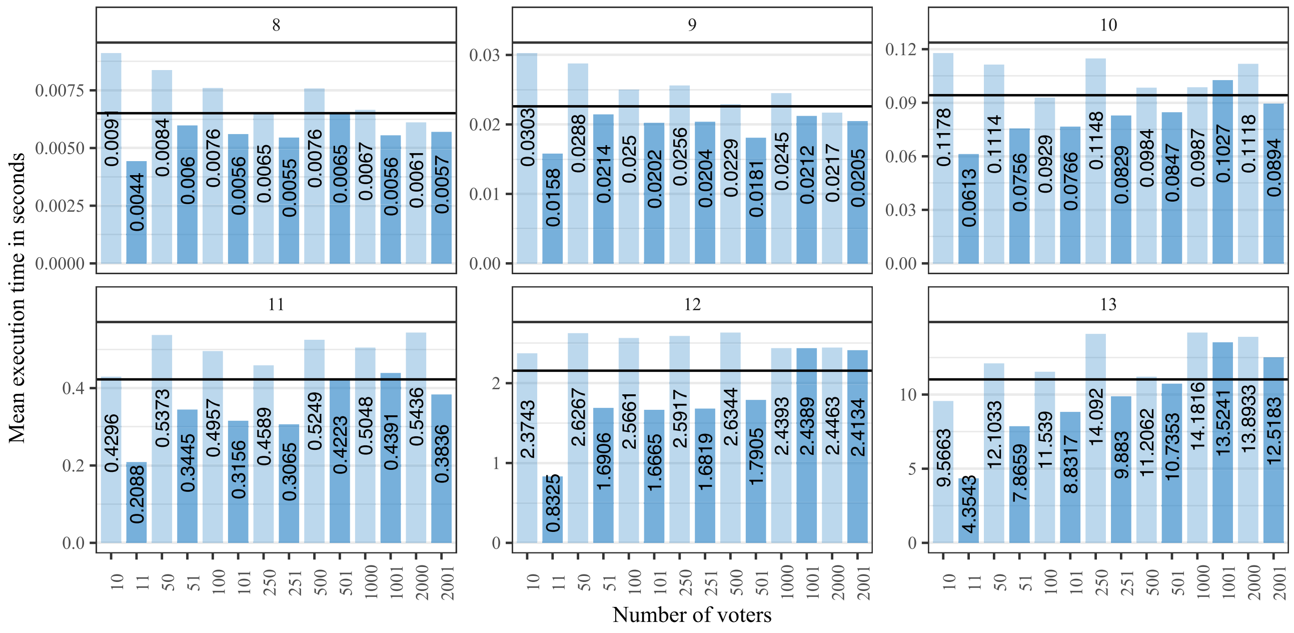


**Fig. 6.** Mean execution time for the 200 profiles of each different combination of number of alternatives and voters.

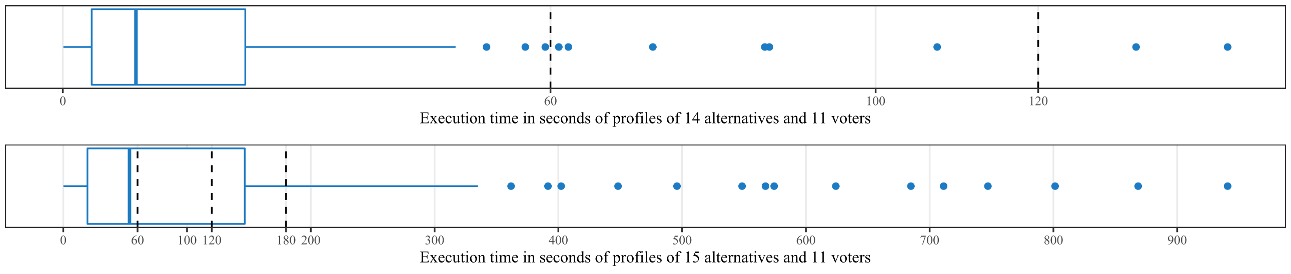


**Fig. 7.** Boxplots illustrating the distribution of the execution time in seconds for 200 profiles of rankings given by 11 voters, for 14 and 15 alternatives. Dashed vertical lines mark the one-minute, two-minute and three-minute thresholds.

value of $m = 11$ voters have been evaluated. The results of the execution time for these profiles are shown in Fig. 7. For 14 alternatives, notice how the execution time on most of the profiles is below the one-minute threshold. However, there are some outliers for which the execution time is considerably larger. For 15 alternatives, the execution time on half of the profiles of rankings is below the one-minute threshold, whereas the execution time on three-fourths of the profiles of rankings is below the three-minute threshold. In this case, we also observe some outliers, leading to an even higher execution time than in the case of 14 alternatives. Overall, this figure hints that the execution time varies largely, even for profiles of rankings with the same number of alternatives and voters. Therefore, there must be other factors that affect the difficulty of determining the winning ranking.

## 5. Factors influencing the execution time

As has been shown in the previous section, the number of alternatives and voters are not the only factors influencing the execution time. Some authors have pointed out that the difficulty of the profile of rankings may have a large influence on the execution time required to determine the winning ranking (Ali & Meila, 2012). However, the measurement of this difficulty and the identification of the factors influencing the execution time are not trivial problems, specially bearing in mind that not all algorithms for the computation of the Kemeny ranking might be affected in the same manner.

For example, in algorithms such as ME, the number of alternatives in the outranking matrix which have $\alpha_i$ = True has a great impact on the execution time, as it dominates the number of recursive calls (which is analogous to the number of solutions explored). For this reason, larger values of alternatives rocket the execution time for the algorithm (Rico et al., 2021b). In the profiles of rankings used in this work, every group of 200 profiles with the same number of alternatives and voters in the dataset show a normal distribution of the number of alternatives that fulfill the property $\alpha_i$ = True. However, the impact of this factor is not so relevant when new prune criteria are added that allow to discard branches even if they have alternatives with this condition, as happen in the two algorithms proposed in this paper.

Another interesting point about the execution time can be observed in the results shown for larger values of $n$ in Fig. 6. This figure illustrates that, although the execution time seems constant for profiles of rankings with an even number of voters, a gentle increasing trend appears for profiles of rankings with an odd number of voters.

Betzler, Fellows, Guo, Niedermeier, & Rosamond (2009) propose to use the average Kendall distance between the rankings in the profile to measure the difficulty to determine a winning ranking. This can be defined in terms of the outranking matrix **O** as follows:

$$\Delta = \frac{1}{h}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} o_{ij}\cdot o_{ji}\,,$$

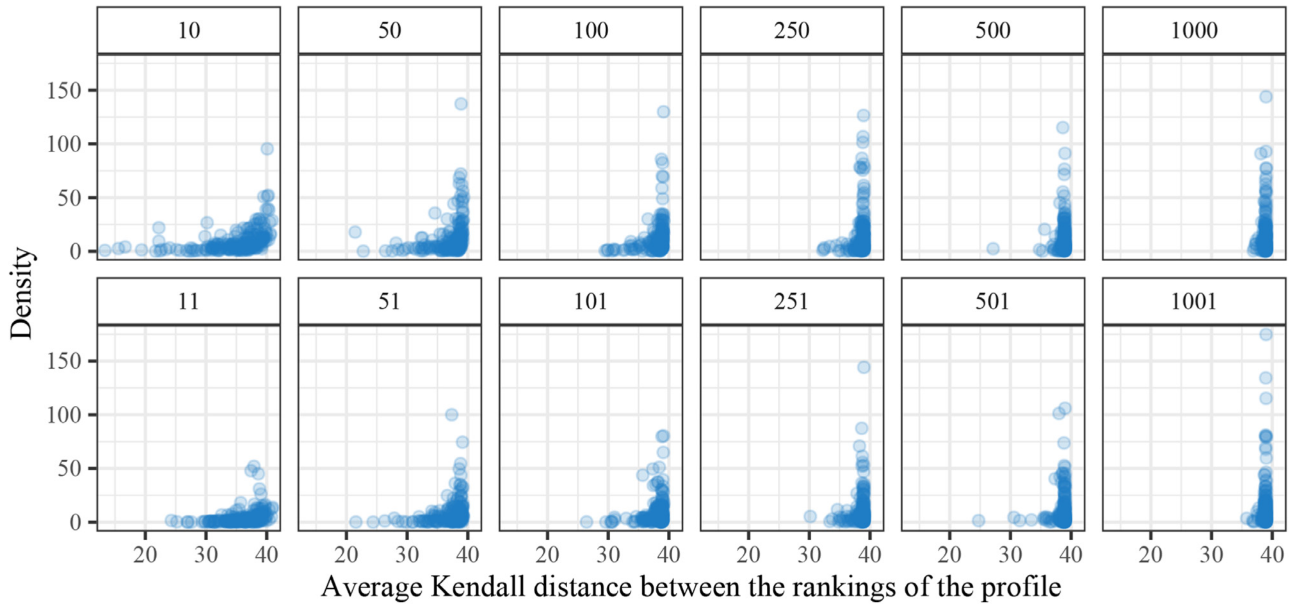


**Fig. 8.** Average Kendall distance ($X$-axis) in comparison to the execution time ($Y$-axis) for the profiles of rankings of 13 alternatives, separated by the number of voters.

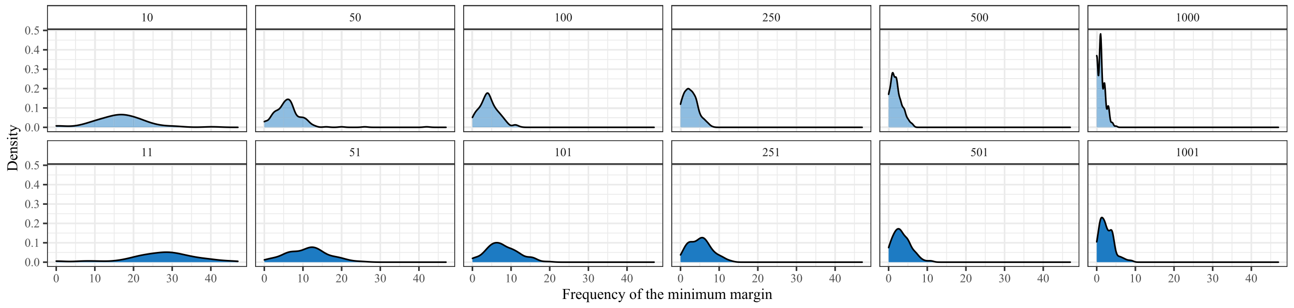


**Fig. 9.** Distribution of the index $\sigma$ (frequency of the minimum margin) for the profiles of rankings with 13 alternatives in comparison to the execution time, separated by the number of voters.

where $h = \frac{n\cdot(n-1)}{2}$ is the triangular number of the number of voters. The value of the average Kendall distance in comparison to the execution time is illustrated in Fig. 8 for the profiles of rankings with 13 alternatives. It can be observed that the profiles of rankings leading to a higher execution time are typically those for which the average Kendall distance is greater. This fact suggests that the disagreement between voters and the execution time may be related to the number of rankings pruned from the search space.

Furthermore, as can be observed both in Figs. 4 and 6, the behavior of the algorithm strongly depends on whether the number of voters is odd or even. In fact, Fig. 4 shows how this different behavior for profiles of rankings with an odd and even number of voters appears when pruning according to the Condorcet winning criterion. A potential explanation for the difference between the profiles of rankings with an odd and even number of voters is the lowest margin that can be obtained in the pairwise comparison. When the number of voters is even, alternatives can be tied in the pairwise comparison and, therefore, the lower margin that can be attained in a pairwise comparison is 0. This implies that for two such alternatives the same points will be added to the distance of the ranking to the profile. For this reason, it would be necessary to explore both alternatives rather than one, as would be the case for an odd number of voters.

Another possibility to measure the disagreement between voters is to consider the index introduced in (Rico, Vela, & Díaz, 2022; Rico, Vela, & Díaz, 2021a). This index measures the number of times that voters have the maximum possible disagreement for a pair of alternatives. This means that half of voters prefer one alternative over the other one, whereas the other half of voters has the opposite preference. This pairwise maximum disagreement is linked to the lowest margin $\mu$ for a pairwise comparison, which equals 0 when $m$ is even and equals 1 when $m$ is odd. The index is formally defined as follows

$$\sigma = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_{ij}\,, \tag{7}$$

where

$$\gamma_{ij} = \begin{cases} 1, & \text{if } |o_{ij} - o_{ji}| = \mu, \\ 0, & \text{otherwise.} \end{cases}$$

The distribution of the index $\sigma$ for each number of voters for the profiles of rankings with 13 alternatives is shown in Fig. 9. The displayed density plot shows a different behavior for even numbers of voters than for the corresponding odd numbers of voters within a unit difference.

## 6. Conclusion

In this paper, we have proposed different exact algorithms for solving the Kemeny problem. The proposed algorithms improve the execution time of the one proposed by Azzini & Munda (2020), one of them even reducing at least an 88% its execution time in average. Furthermore, this improvement appears to be even more pronounced as the number of alternatives increases. The best of the proposed algorithms guarantees an assumable execution time up to $n = 14$ alternatives.

Moreover, some factors potentially impacting the execution time have been studied. Special focus has been given to some aspects of the profiles of rankings that may have an impact on the execution time. Further research along this direction may be of interest, as we have provided empirical evidence that hints that profiles of rankings with the same number of alternatives and voters might differ greatly in their execution time, specially for larger numbers of alternatives. The identification of these factors would allow to parametrize the algorithm and further refine the study of the execution time for profiles of rankings with a greater number of alternatives.

A possible extension of this topic includes the development of parallel algorithms or the exploitation of approximate metaheuristic techniques with the aim of speeding up the execution time, which is often necessary in many real-life applications.

## References

Ali, A., & Meila, M. (2012). Experiments with Kemeny ranking: What works when? *Mathematical Social Sciences, 64*(1), 28–40.

Amodio, S., D'Ambrosio, A., & Siciliano, R. (2016). Accurate algorithms for identifying the median ranking when dealing with weak and partial rankings under the Kemeny axiomatic approach. *European Journal of Operational Research, 249*(2), 667–676.

Arora, S., & Barak, B. (2009). *Computational complexity: A modern approach*. Cambridge University Press.

Arrow, K., & Raynaud, H. (1986). *Social choice and multicriterion decision-making* (1st). The MIT Press.

Azzini, I., & Munda, G. (2020). A new approach for identifying the Kemeny median ranking. *European Journal of Operational Research, 281*, 388–401.

Bailey, R. W. (1998). The number of weak orderings of a finite set. *Social Choice and Welfare, 15*(4), 559–562.

Barthelemy, J. P., Guenoche, A., & Hudry, O. (1989). Median linear orders: Heuristics and a branch and bound algorithm. *European Journal of Operational Research, 42*, 313–325.

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

Betzler, N., Fellows, M. R., Guo, J., Niedermeier, R., & Rosamond, F. A. (2009). Fixed-parameter algorithms for Kemeny rankings. *Theor. Comput. Sci., 410*(45), 4554–4570.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (Eds.). (2016). *Handbook of computational social choice*. Cambridge University Press.

Cao, Z., Guo, Y., Ao, Y., & Zhou, S. (2020). Dysregulated micrornas in laryngeal cancer: A comprehensive meta-analysis using a robust rank aggregation approach. *Future Oncology, 16*(33), 2723–2734.

Emond, E. J., & Mason, D. W. (2002). A new rank correlation coefficient with application to the consensus ranking problem. *Journal of Multi-Criteria Decision Analysis, 11*(1), 17–28.

Fishburn, P. C. (1973). *The theory of social choice*. Princeton: Princeton University Press.

Goerlich, F. J., & Reig, E. (2021). Quality of life ranking of spanish cities: A non-compensatory approach. *Cities, 109*, 102979.

Hemaspaandra, E., Spakowski, H., & Vogel, J. (2005). The complexity of Kemeny elections. *Theoretical Computer Science, 349*(3), 382–391.

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

Muravyov, S. V. (2007). Rankings as ordinal scale measurement results. *Metrology and Measurement Systems, 14, nr 1*, 9–23.

Muravyov, S. V. (2013). Ordinal measurement, preference aggregation and interlaboratory comparisons. *Measurement, 46*, 2927–2935.

Oliveira, S. E., Diniz, V., Lacerda, A., Merschmanm, L., & Pappa, G. L. (2020). Is rank aggregation effective in recommender systems? An experimental analysis. *ACM Transactions on Intelligent Systems and Technology (TIST), 11*(2), 1–26.

Rico, N., Vela, C. R., & Díaz, I. (2022). Runtime bounds prediction for the Kemeny problem. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-022-03881-2.

Rico, N., Vela, C. R., & Díaz, I. (2021a). An analysis of the indexes measuring the agreement of a profile of rankings. In Proceedings of the xix conference of the spanish association for artificial intelligence (Caepia) (pp. 192–197).

Rico, N., Vela, C. R., Pérez-Fernández, R., & Díaz, I. (2021b). Reducing the computational time for the Kemeny method by exploiting Condorcet properties. *Mathematics, 9*(12), 1380.

Roberts, F., & Tesman, B. (2009). *Applied combinatorics, second edition* (2nd). Chapman Hall CRC.

Schulze, M. (2011). A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare, 36*(2), 267–303.

Tideman, T. N. (1987). Independence of clones as a criterion for voting rules. *Social Choice and Welfare, 4*(3), 185–206.

Young, H. P. (1988). Condorcet's theory of voting. *American Political Science Review, 82*(4), 1231–1244.