Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-022-03881-2

ORIGINAL RESEARCH

Runtime bounds prediction for the Kemeny problem

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

Received: 5 August 2021 / Accepted: 27 April 2022

Abstract
The time required for solving the ranking aggregation problem using the Kemeny method increases factorially with the number of alternatives to be ranked, which prevents its use when this number is large. Exact algorithms use domain information to discard rankings as possible solution, thus saving runtime. The amount of rankings that can be discarded varies for each profile and cannot be known beforehand. For profiles of rankings with large number of alternatives, the amount of rankings discarded highly affects the feasibility of the computation of Kemeny ranking. How to identify the profiles that are more time-consuming when finding the Kemeny ranking is not trivial. In this work we propose the use of machine learning models to predict how difficult is to obtain the Kemeny ranking in terms of runtime. The results obtained are promising, with values of the area under the curve metric over 80%. Furthermore, it is possible to extract from the proposed models the characteristics of the profile of rankings that impact on the runtime.

Keywords Supervised learning · Runtime · Ranking aggregation · Kemeny method

1 Introduction

The options to aggregate rankings over a set of alternatives in order to determine the *consensus ranking* that best summarizes the information has been deeply studied in the field of social choice theory (Fishburn 1973).

Many *consensus ranking* methods are based on the so-called Condorcet rules (Condorcet 1785). A *Condorcet winner* of an election is the alternative that beats all other alternatives in a pairwise comparison by a majority of the votes. However, such alternative may not exist. Furthermore, this problem expands if the aim is to determine not only the winning alternative but also to establish a winning ranking. In this case, an alternative should be ranked at a better position than another alternative in the final ranking whenever the former defeats the latter by more than half of the votes. These majority relations obtained from the pairwise comparison of the alternatives are not necessarily transitive (Arrow and Raynaud 1986).

It is usual to find in practice profiles (ranking sets) with cycles, therefore making impossible the use of the Condorcet rule. Several attempts to fully understand and axiomatize Condorcet's approach for solving the problem in the presence of cycles have been proposed (Schulze 2011; Kemeny 1959; Pérez-Fernández et al. 2016). They are usually referred as *Condorcet methods*, since the Condorcet winner is returned as winner in case that it exists but they also provide a solution in presence of cycles. Among this family of methods, the one proposed by Kemeny not only ensures the winner to be the Condorcet winner if it exists, but also to retrieve the Condorcet ranking as the only solution in case that a Condorcet ranking exists and to provide a solution in the presence of cycles. It is the only ranking aggregation method that is neutral, consistent and Condorcet at the same time (Young and Levenglick 1978; Hemaspaandra et al. 2005). Due to these properties the method is appropriate to be used in many situations. However it is an NP-hard (Bartholdi et al. 1989) problem, which makes it not suitable for its use in most of the real situations, especially in presence of time constraints.

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

✉ Irene Díaz
sirene@uniovi.es

Noelia Rico
noeliarico@uniovi.es

Camino R. Vela
crvela@uniovi.es

[1] Computer Science Department, University of Oviedo, Oviedo, Spain

Published online: 17 May 2022

The computation of ranking aggregation methods (Brandt et al. 2016) has gained attention in recent years, as they arise quite commonly in real-life problems (Oliveira et al. 2020). Recently, (Azzini and Munda 2020) introduced an exact algorithm that greatly improved the runtime in relation to others published before. However, the runtime of this algorithm still varies widely for profiles defined for the same number of alternatives and voters. An implementation of this algorithm for profiles of rankings with 12 candidates and constant number of voters is shown in Rico et al. (2021b), with runtimes ranging from values under 5 s to values slightly higher than 4000 s. Notice that, although there is an obvious dependence between the runtime and the number of alternatives of the profile, the nature of branching algorithms makes that even for problems with the same size the real execution of the algorithm is sometimes possible and sometimes not, as it depends on the solutions pruned from the search space, which cannot be determined prior to the execution itself. Some authors have previously pointed the attention to the fact that the number of alternatives does not completely determine the runtime required by different algorithms (Ali and Meila 2012; Betzler et al. 2009) to reach the consensus ranking. Therefore, it is important to study those features of the profile that influence the behavior of the algorithms. Otherwise, when new algorithms are proposed to solve this problem, comparisons could be unrealistic if their behavior cannot be determined a priori based on the profile characteristics.

This behavior might indicate that some aspects as, for example, the disagreement between the voters in the profile of rankings, may have an impact on the cost to find the solution to the problem, which would mean that some profiles are more *difficult* than others. How to determine this difficulty is not trivial and remains undefined. In this paper *difficulty* is defined as the amount of time required to find the solution of the Kemeny problem. To the best of our knowledge, this is the first time that a machine learning-based approach has been used to predict the difficulty of finding the Kemeny ranking from a given profile of rankings.

Therefore, the aim of this work is two fold. One the one hand, to identify potential characteristics of a profile that make more difficult the obtaining of the Kemeny ranking. On the other hand, to explore a machine learning approach using these characteristics to predict whether a given a profile would require a large runtime to determine the Kemeny ranking using a certain aggregation algorithm. This is important from a research point of view as, sometimes, the runtime required to find the Kemeny ranking is unacceptable. To sum, given a profile of rankings this work aims to answer the question *how difficult is to find its solution*.

In order to answer the stated question, different datasets and machine learning models will be provided in order to study different behaviors of runtime for this complex problem. The novelty of this work is based on (1) the use of a machine learning approach to study the problem, (2) the characterization of the profiles by some indices and (3) the construction of benchmarking data sets publicly available.

The remainder of this paper is organized as follows. Section 2 puts in context the Kemeny method and the exact algorithm used as baseline of this work. The indices proposed to characterize the profiles are reviewed in Sect. 3. The profile of ranking gathering is detailed in Sect. 4 and the machine learning approaches are further explained in Sect. 5. Final conclusions are outlined in Sect. 6.2.

2 Algorithm to determine the Kemeny ranking

Let us refer to a profile of rankings π_m^n that contains the preferences given by m voters in the form of rankings over a set of alternatives $\mathcal{A} = \{a_i, \ldots, a_n\}$. The rankings in π_m^n are transitive complete orders that defines strict $a_i \succ a_j$ or weak $a_i \sim a_j$ relationships for every pair of alternatives in $\mathcal{A}$. When the profiles are compared pair-wisely, a profile can be represented by means of an outranking matrix $\mathbf{O}$ of dimension $n \times n$ (Arrow and Raynaud 1986), where each element $o_{ij} \in \mathbf{O}$ in the i-th row and j-th column represents the number of times that the element $a_i \succ a_j$ or $a_i \sim a_j$, by annotating 1 point every time that the first situation happens and $\frac{1}{2}$ points for the second. In the case described, as the profile contain only complete rankings, this matrix fulfils the constant sum property $o_{ij} + o_{ji} = m$.

Kemeny (1959) defines the distance between any two rankings r_i and r_j on the set of alternatives $\mathcal{A}$ 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 their order in both rankings. Concretely, he proposes to annotate 2 points when the alternatives are in different order and 1 point if they are tied only in one of the rankings. Using this, the distance from one ranking r_i to the profile π_m^n is the sum of d_{r_i,r_j}, $\forall r_j \in \pi_m^n$. The Kemeny method uses this distance in order to compare the rankings, by evaluating all the possible rankings of the set $\mathcal{A}$ in order to determine the one (or ones) that is the closest to the profile, which is the *Kemeny ranking*. Note that the distance given by Kemeny is equivalent in relative terms to obtain a distance (known as Kendall distance (Kendall 1938)) from a ranking r to a profile using its outranking matrix $\mathbf{O}$ such that $\delta(r, \pi_m^n) = \sum_{i=1}^{n} \sum_{j=1}^{n} o_{ij} \cdot x_{ij}$, *with* $x_{ij} = 0$ if $a_i \succ^r a_j$ and 1 otherwise. In Azzini and Munda (2020) an exact algorithm to solve the Kemeny problem is proposed. It is based on reducing the number of tentative solutions to explore as possible Kemeny ranking, which initially would be all the possible permutations of $\mathcal{A}$, by using a necessary condition that the winner of the Kemeny ranking must fulfill.

They proved that the alternative $a_i \in \mathcal{A}$ at the top position of the winning ranking must satisfy $\sum_{j=1}^{n} o_{ij} \geq \sum_{j=1}^{n} o_{ji}$. For the sake of simplicity, α_i denotes the truth value obtained from this Boolean expression. Therefore, the alternative a_i at the first position of the ranking must satisfy $\alpha_i = True$, which allows to considerably reduce the set of rankings to explore as possible solutions. Let us emphasize that this gives a necessary but not sufficient condition. Nevertheless, it allows to avoid the exploration of many of the rankings as potential solution, considerably reducing the runtime in relation to other algorithms. Furthermore, this can be recursively applied to any subset of $\mathcal{A}$, also reducing the search space of possible solutions. The idea is to build rankings alternative by alternative, by adding to a prefix ρ only alternatives that have $\alpha_i = True$, and then to remove the alternative a_i from the matrix and repeat the process in the recursive call. This recursive approach makes impossible to determine in advance the number of rankings that will be discarded as possible solutions, which have a great impact on the runtime. In Rico et al. (2021b) some considerations over this proposal are introduced, obtaining the ME algorithm, which is detailed in Algorithm 1.

Algorithm 1 ME algorithm

1. Define an empty list of tentative solutions σ.
2. Take $\mathbf{O}$ and define the set $\eta = \{a_i \in \mathbf{O}/\alpha_i = True\}$.
3. Initialize a stack of prefixes to explore. Create one prefix for each alternative $a_i \in \eta$.
4. Take (and remove) the first prefix ρ from the stack.
5. Consider the submatrix $\mathbf{O}_\rho$, which contains only the alternatives of the set $\mathcal{A}$ that are not already fixed in the prefix ρ.
   - If $\mathbf{O}_\rho$ dimension 2×2, then for the two remaining alternatives a_i, a_j:
     - If $o_{ij} > o_{ji}$, add $\rho \succ a_i \succ a_j$ to σ.
     - If $o_{ji} > o_{ij}$, add $\rho \succ a_j \succ a_i$ to σ.
     - If $o_{ij} = o_{ji}$, add both $\rho \succ a_i \succ a_j$ and $\rho \succ a_j \succ a_i$ to σ.
   - If the number of alternatives in $\mathbf{O}_\rho$ is $n \geq 3$, determine

     $$\eta = \{a_i \in \mathbf{O}_\rho/\alpha_i = True\}.$$

     Add one prefix of the form '$\rho \succ a_i$' to the stack for each $a_i \in \eta$.
6. Repeat *Step 4* and *Step 5* until η is empty.
7. Compute $\delta(s_i, \pi_m^n)$ for all the rankings $s_i \in \sigma$.
8. Compute $\delta_{min} = \min_{s_i \in \sigma}(\delta(s_i, \pi_m^n))$.
9. Delete from σ all rankings such that $\delta(s_i, \pi_m^n) > \delta_{min}$.

3 Indices to measure the difficulty of the profile

The different number of rankings that can be discarded in the above introduced recursion process influences the runtime required by Algorithm 1 to find the Kemeny ranking, which makes the runtime to variate even for profiles with the same number of alternatives and voters. Intuitively, the distribution of the votes over the alternatives could have an impact on the uncertainty to determine the final winning ranking, as it is natural to think that the closer the opinion of the voters the easier to find the consensus given by the Kemeny ranking. Unfortunately, how the uncertainty in the profile can be measured and how it affects to the difficulty of solving the problem is not defined. This definition is not a trivial matter. For example, the ranking of Condorcet defines an ideal profile without cycles that could be considered a goal. However, this not necessarily represents the *less difficult* approach in computational terms, as the difficulty exclusively depends on the algorithm used to solve the problem. Despite some authors have mentioned this fact, there is not a universal measure to weight how difficult the problem to solve is (Ali and Meila 2012; Betzler et al. 2009). In a previous work (Rico et al. 2021a), we have reviewed and proposed new indices in order to characterize different aspects of the profile. From this, it has been concluded that, although some of the indices present different behavior in relation to the runtime, none of them shows a clear influence to determine on its own the time required by the algorithm to determine the winning ranking for a concrete profile. In the current paper the intervals within each of these indices ranges are also studied, with the aim of being able to make them comparable for profiles with different n and m. Both indices and boundaries are listed in Table 1 and briefly described in the following.

- Range of the indices. As shown in Table 1, the proposed indices vary in a range that depends either on n, m or the corresponding number of pairwise comparisons of the alternatives τ_n or rankings τ_m. This means that, if

Table 1 Summary of the indices defined in this paper (separated in subsections) used for characterizing different aspects of the profile. This table shows the description of the index as well as the interval of values that they can take

Index	Description	Formula	Interval
μ_1	Average Kendall distance	$\frac{1}{\tau_m}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} o_{ij}\cdot o_{ji}$	$[0, k\cdot\tau_n], k=\begin{cases}\frac{m}{2(m-1)}, & m>2\\ 1, & \text{otherwise}\end{cases}$
μ_2	Maximum range of alternative positions	$\max_{r_i,r_j\in\pi_m^n, a_k\in\mathcal{A}}(\lvert r_i(a_k)-r_j(a_k)\rvert)$	$[0, n-1]$
μ_3	Minimum range of alternative positions	$\min_{r_i,r_j\in\pi_m^n, a_k\in\mathcal{A}}(\lvert r_i(a_k)-r_j(a_k)\rvert)$	$[0, n-1]$
μ_4	Average range of alternative positions	$\frac{1}{n}\sum_{a_k\in\mathcal{A}}(\max_{r_i\in\pi_m^n} r_i(a_k)-\min_{r_j\in\pi_m^n} r_j(a_k))$	$[0, n-1]$
μ_5	Bound to the optimal cost	$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\max(o_{ij},o_{ji})-\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\min(o_{ij},o_{ji})$	$[x\cdot\tau_n, m\cdot\tau_n]x=m \bmod 2$
μ_6	Number of dirty triplets	$\frac{1}{3}\sum_{i=1}^{n}\sum_{j\neq i}\sum_{k\neq i\neq j} y_{ijk}, y_{ijk}=\begin{cases}1, & o_{ij},o_{jk},o_{ki}>h\\ 0, & \text{otherwise}\end{cases}$	$[0, \frac{n!}{3!(n-3)!}]$
μ_7	Mode margin	$\text{argmax}_{M\in\mathbb{M}}\#\{(i,j)\mid 1\le i<j\le n, M=\ell_{ij}\}$	$[min_\ell, m]$
μ_8	Number of different margins	$\#(\mathbb{M})$	$[1, \min(\tau_n, \frac{m-min_\ell}{2}+1)]$
μ_9	Frequency of the minimum margin	$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\gamma_{ij}$ with $\gamma_{ij}=\begin{cases}1, & \ell=\min_\ell\\ 0, & \text{otherwise}\end{cases}$	$[0, \tau_n]$
μ_{10}	Distance to Borda ranking	$\delta(b,\pi_m^n)$	$[0, \tau_n\cdot m)$
μ_{11}	Number of overall preferred alternatives	$\sum_{a_i\in\mathcal{A}}\alpha_i$	$[1, n]$
μ_{12}	Standard deviation of $\boldsymbol{\theta}$	$sd(\boldsymbol{\theta})$	$[0, \tau_n\cdot m)$
μ_{13}	Standard deviation of $\boldsymbol{\beta}$	$sd(\boldsymbol{\beta})$	$[0, \tau_n]$
μ_{14}	Mean of $\boldsymbol{\beta}$	$mean(\boldsymbol{\beta})$	$[0, n-1)$
μ_{15}	Median of $\boldsymbol{\beta}$	$median(\boldsymbol{\beta})$	$[0, n-1)$
μ_{16}	Difference between mean and median of $\boldsymbol{\beta}$	$\lvert\mu_{14}-\mu_{15}\rvert$	$[0, n-1)$

the aim is to use these indices to compare profiles with different number of alternatives and voters, they must be normalized using the intervals defined in the fourth column of the table. Further insight about how these ranges are obtained is given below.

- Based on Kendall distance (μ_1) (Betzler et al. 2008; Kendall 1938) it is computed the averaged Kendall distance between every pair of rankings in the profile. This distance depends only on the number of alternatives. One ranking and its opposite have the furthest possible distance, τ_n, as every pair of alternatives add one point to the computation of the distance. Thus, the upper bound for Kendall distance cannot exceed the value τ_n. This worst scenario happens when there are two voters in the profile expressing their opinions by opposite rankings. On the other hand, if the m voters of a profile express the same preference, the distance between each pair of rankings is trivially 0 and that is the lowest value μ_1 can take.
- Based on the range the different positions an alternative takes in a profile is important to select it in a certain position in the Kemeny ranking obtained by Algorithm 1. For this reason we consider the minimum (μ_2), maximum (μ_3) and average (μ_4) position ($r_i(a_k)$) that each alternative a_k can take in a ranking r_i. These indices vary between 0 (each alternative takes the same position in all the rankings) and $n-1$ (at least an alternative is in the best position in one ranking and in the worst position in another one). A high value of the maximum range means that there is at least one alternative for which the voters do not have a clear opinion meanwhile a low value for the minimum range means that there is at least one alternative for which the voters have a clear opinion.
- Based on computing a bound to the optimal cost. The maximum distance of a ranking to a profile (μ_5) can be computed using τ_n elements of the outranking matrix. Thus, the furthest ranking is obtained the profile contains just one ranking. The minimum distance of a ranking to the profile would happen if those τ_n elements considered have the minimum value, which is 0 for profiles with m even and 1 for profiles with m odd. When the bound to the optimal cost is small, all the rankings would have similar distances so it is likely that more rankings need to be explored in order to determine the Kemeny winner.
- Based on the number of dirty triplets (i.e. intransitive cycles of length three) that measure the agreement among the voters as the outranking matrices when there is a Condorcet ranking do not contain cycles. These could lead to a situation in which the Kemeny ranking for the profile requires more time to be found.

- Based on the *margin* of the voters preferring one alternative a_i over another one a_j is $|o_{ij} - o_{ji}|$. It indicates the global preference expressed in the profile for one of the two alternatives over the other one, thus it could guide us in the identification of profiles more or less demanding according to algorithm 1. The margin set $\mathbb{M} = \{\ell_{ij} = |o_{ij} - o_{ji}| \mid 1 \leq i < j \leq n\}$ provides useful information about the voters agreement. Each $\ell_{ij} \in [min_\ell, m]$ as the largest margin is given when all the voters prefer one alternative over another. However, the lowest margin depends on whether the number of voters is even or odd (min_ℓ is 1 if m is odd and 0 otherwise). The most frequent margin (μ_7), margin diversity (μ_8, that measures the different global preferences) or the frequency of the minimum margin (μ_9, that indicates if the uncertainty among the voters of the profile is high) are considered. It is obvious by definition that $\mu_7 \in [mod(m, 2), m]$. Index μ_8 depends on the number of voters and on the number of unique rankings in the profile (at most τ_n). The frequency of the minimum margin is at most μ_8.
- Based on Borda count (μ_{10}), Borda (1781) that ranks the alternatives taking into account the number of times that they are ranked better than other alternatives. More specifically, the Borda score α_i of each alternative a_i is computed as $s(a_i) = \sum_{j=1}^{n} o_{ij}$. After these are obtained, the alternatives are sorted by decreasing order of their scores to obtain the consensus ranking b. As two alternatives may have the same Borda score, some alternatives might be tied in the obtained consensus ranking. The distance from this ranking to a profile can be, as for any other ranking, in the interval $[0, \tau_n]$. The maximum distance between a ranking and a profile would be $\tau_n \times m$ in case that the all the voters of the profile give the same ranking and the distance would be considered to the opposite ranking. Nevertheless notice that, as the Borda ranking is obtained from the profile, it will never reach the worst scenario as it is guaranteed not to give the opposite order of all the rankings in the profile. When the voters greatly agree in the preferences of the alternatives, the Borda ranking is closer to the Kemeny ranking and therefore the distance from the Borda ranking to the profile should be lower than in cases when there are more disagreement. For this reason, this index is considered helpful in this framework.
- Based on the row sum of $\mathbf{O}$. Two indices are considered, the number of a priori preferred alternatives (μ_{11}), defined in terms of α_i and the standard deviation of the row sums of $\mathbf{O}$ (μ_{12}) defined in terms of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$, with $\theta_i = \sum_{j=1}^{n} o_{ij}$. As the sum of all the elements of $\boldsymbol{\theta}$ is constant, if all the voters agree in the same ranking, then there is at least one row with m votes and another with 0. On the contrary, if all the alternatives are tied in their pairwise comparison this would yield to all the elements of $\boldsymbol{\theta}$ to have the same value. Therefore indices based on this vector may give an orientation of how the opinion of the voters varies. The maximum number of alternatives with $\alpha_i = True$ cannot exceed the number of alternatives itself. As $\alpha_i = True$ for at least one alternative a_i (Azzini and Munda , 2020), the minimum of value of this index is 1. The standard deviation of the rowsums is at most $\tau_n \cdot m$ by construction.
- Based on the distribution of preferred alternatives, that are studied in terms of the vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$, with $\beta_i = \sum_{j=1}^{n} b_{ij}$ and $b_{ij} = 1$ if $o_{ij} > o_{ji}$ and 0 otherwise. It is straightforward that the indices based on the $\boldsymbol{\beta}$ (μ_{13} to μ_{16}) ranges between 0 and $n - 1$. As this vector contains all the integer numbers in $[0, n - 1]$ only once when the profile has a Condorcet ranking, different variations in relation to this might be related to the increasing of the runtime.

4 Gathering the profiles of rankings

The runtime required by Algorithm 1 to find the Kemeny ranking is not the same for all the profiles, even if the number of alternatives is the same. The aim of this work is to predict using machine learning and the characteristics of the profile if the algorithm can find the Kemeny ranking in an assumable time. The first task is then to obtain a representative dataset to study this problem. In this case profiles with $n \in \{8, 9, 10\}$ alternatives and $m \in \{10,\ 50,\ 100,\ 250,\ 500,\ 1000,\ 2000\}$ voters have been considered in order to build a dataset. Also the same numbers of voters plus 1 have been considered, as initial experiments suggest that whether the number of voters is even or odd influences on the runtime (Rico et al. 2021a). For each combination of (n, m), 200 different profiles have been created. The values of n have been selected due to their acceptable computational time, as they are neither too small to present randomness in the runtime nor too large in relation to time constraints. Each profile has been created according to the following steps:

1. Randomly select the number $d \leq m$ 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 add this to the dataset.

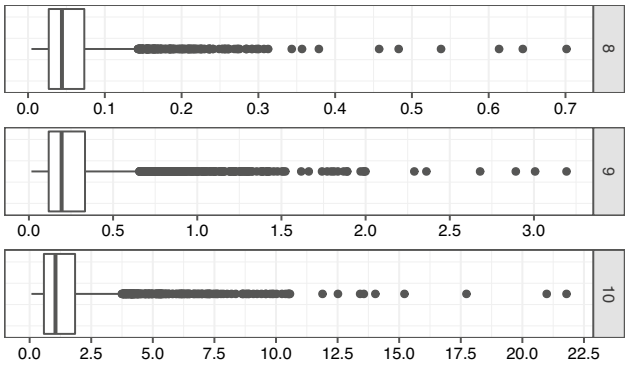

Fig. 1 Distribution of the runtimes in seconds required by the profiles of the data set build, grouped by their number of alternatives (8 top, 9 middle, 10 bottom)

The last step is included to avoid profiles whose runtime can be reduced using additional information of the domain, as in presence of a Condorcet winner it is not necessary to explore the remaining alternatives (Rico et al. 2021b).

The runtime associated to each profile is computed in order to build the variable to predict and study it in relation to the characteristics of the profile measured by the indices. Runtimes have been obtained using a Python 3.8 implementation of Algorithm 1. In order to minimize the impact of other processes that might be being executed by the processor on the measurement of the runtimes, the algorithm has been used three times with each profile and the median value has been considered.

Note that runtimes do not depend only on the number of alternatives (Fig. 1). In addition, it is impossible to predict exact runtimes for profiles of rankings with numbers of alternatives not used in the training set. Thus, instead of modeling the problem as a regression one, we focus ourselves on two problems. The first one is based on identifying those profiles that require an abnormally high runtime to find Kemeny's ranking. With this identification, it is possible to avoid them. The second one is based on identifying those profiles that require the shortest possible runtime to find Kemeny's ranking.

5 Identification of affordable profiles of rankings

Supervised machine learning methods learn a mapping from an input to an output based on available observations (Kubat 2017). Each observation is therefore a pair with an input vector and its corresponding output. The model produced predicts this output from the input data and can be later used for mapping new examples into an unknown output. Binary classification methods are proposed to identify the previous stated profiles. In the following, we explain how the models are obtained.

5.1 Training the models

Classification models presented in this work are trained by means of CART decision trees and random forests (Kubat 2017). These methods are suitable for this problem as they are explainable and one of the goals of the work is to characterize the profiles. Using explainable machine learning methods make easier to explain the impact of the input variables on the outcome. Moreover, other algorithms have been studied in a preliminary phase for solving this problem and both the decision trees and random forests show competitive results against them. First of all, different parameter sets for each algorithm are defined to look for the best parameter configuration of each method, this is, the values of the parameters of the algorithm that lead to the best models of the data set. These parameters depend on the algorithm and are the *number of trees* for random forest and the *depth of the tree* and *minimum number of objects in the leaves* for the CART trees. Then, for each classification algorithm and the different parameter settings of the classification algorithm, all the models are trained using a training-test validation procedure, the input dataset is divided into training and test set according to a 80–20% split. The training set is used to build the model using repeated 10-cross-validation (5 times). Due to the imbalanced nature of the problem (explained in Sects. 5.4, 5.5), the models have been trained using a down-sampling technique (as it has been proved to yield better results than other techniques such as up-sampling and ROSE (Lunardon et al. 2014) with this data).

Algorithm 2 Training process

- Define sets of model parameter values to evaluate.
- For each parameter set.
  - For each resampling iteration.
    - Hold-out specific samples.
    - Fit the model on the remainder.
    - Predict the hold-out resamples.
- Determine the optimal parameter set
- Fit the final model to all the training data using the optimal parameter set

5.2 Model evaluation

Classification models are evaluated by means of measures based on the confusion matrix (Kubat 2017) associated to a binary classification problem where the positive class is the minority class. The Area Under the ROC Curve (AUC) (Kubat 2017) is also used for evaluating the models. This measure represents the true positive rates against the false positive rates for different thresholds of classification. However, the riskier situation occurs if a profile is wrongly identified as affordable in terms of runtime, which would mean an unexpected increase of the runtime when trying to find the Kemeny ranking. For this reason it is also interesting to track the *sensitivity* (also known as recall) that measures the amount of true positives correctly classified in relation to the total amount of positives in the dataset. High values of this measure leads to models that correctly classify instances from both the majority and the minority classes. Models will be trained focusing on the maximization of AUC.

The next two subsections show the performance of the models when different datasets are considered.

5.3 Identifying profiles with abnormally high runtime

In this subsection, we focus on detecting those profiles to avoid, i.e, profiles whose runtime to obtain Kemeny ranking is very high. In addition, it is important to characterize them in terms of the indices described in Sect. 3.

The data set to perform this task is built using the profiles introduced in Sect. 4. Note that when the number of alternatives is fixed, most of the profiles behave similarly in terms of runtime when using Algorithm 1. However, as it was highlighted in previous section, there are some profiles whose runtime is abnormally high in relation to those with the same number of alternatives (see Fig. 1). A profile with a runtime very high is here called *outlier*. It is considered that a runtime is very high if it is greater than $Q3 + 1.5 \cdot IQ$ (being IQ the interquartile range shown by the boxes in Fig. 1, and Q3 the third quartile). Using this information, a dataset is generated where each profile is characterized by the indices introduced in Sect. 3. The variable to predict is binary (whether the profile is an *outlier* or not). Formally defined, each observation of the dataset has the form $x = \{\mu_1, \ldots, \mu_{16}\}$ and an y takes two possible values (*yes* if it is an *outlier* and *no* otherwise). Note that the percentages of outliers (y=*yes*) obtained for the subsets with 8, 9 and 10 alternatives are 6.43, 7.89 and 7.39% respectively. This means that this dataset is highly imbalanced.

In order to make the problem independent of the number of alternatives of the profiles used for training the models, this dataset is normalized using the ranges defined in the last column of Table 1.

We have firstly trained the models introduced in Sect. 5.1 considering 4 different datasets, the one containing the profiles with size $n = 8, 9$ or 10, and each subset of this one fixing n. Independently of the dataset, the best performance was obtained by random forest methods. Table 2 shows the results of the AUC and sensitivity measures of the models trained using the random forest algorithm. The values of AUC are always greater than 0.8. Furthermore, the sensitivity of the model has a soft increase when the number of alternatives is bigger, and also when all the dataset is used for training the model and therefore more information is available. Thus, the results in terms of performance are acceptable.

As previously mentioned, the purpose of this work is not only to classify the profiles but also to identify the indices that impact on the increment of the runtime required by Algorithm 1 in order to find the Kemeny ranking. As the method that obtains the best performance is random forest, it is possible to obtain variable importance in terms of the impurity of the dataset after splitting using a variable (Greenwell and Boehmke 2020). For all the trained models, the top 5 most important variables are (1) μ_{11}, (2) μ_{12}, (3) μ_1, (4) μ_{13} and (5) μ_{10}.

We have found that index μ_{11} is the most important one. Thus, it is clear that this number has a large impact in the number of rankings evaluated, as greater amounts of them can be discarded from the exploration when this index is low. Note that although the theoretical complexity of the algorithm does not change (as the problem of finding the Kemeny ranking is NP-hard), the reduction of the search space of possible solutions implies the feasibility of the execution of the algorithm when the number of candidates increases. As shown in Algorithm 1, only the rankings with a candidate c_i that has an associated $\alpha_i = True$ in the first position must be considered as possible solution. This means that the search space with size $n!$ is reduced ensuring that it is in the worst case $n! - (\mu_{11} \cdot \frac{n!}{n})$, which ensures a shorter execution time. Also, μ_{12} and μ_{13} provide useful information to predict the outcome. These indices are related to the distribution of the votes so we consider very relevant this finding as it is highlighted that profiles of rankings with the same numbers of voters can behave very different as we supposed. The average Kendall distance μ_1 appears also in a high position. The distance of the Borda ranking to the profile has also impact as when there is more agreement

Table 2 Results on the test set for different subsets of the data

	n = 8	n = 9	n = 10	all
AUC	0.8313	0.8797	0.8204	0.8636
Sensitivity	0.7500	0.7955	0.8293	0.8429

Borda and Kemeny rankings are closer. When we test the method for more complex profiles we obtain that the more important indices are the same initially obtained, reinforcing the initial hypothesis.

5.4 Generalization for more complex profiles

The experiments confirm that there are some characteristics of the profile that can be measured by indices with impact on the runtime required to obtain Kemeny ranking. As these indices are normalized to make them independent on n and m, a model trained with the above-defined procedure can be also used to predict if the runtime is assumable for a profile with a different number of alternatives n than the ones used to train the model. Thus, let us consider now a fifth training set only composed by profiles with $n = 8$ or $n = 9$ alternatives and a test set with profiles with $n = 10$ alternatives. Random forest also performs better than decision trees in this case, obtaining an AUC of 0.8173 and a sensitivity of 0.8164 in the test set. The top five features are a permutation of the obtained in the previous experiment: (1) μ_{11}, (2) μ_{13}, (3) μ_{12}, (4) μ_1 and (5) μ_{10}, reinforcing their importance.

This experiment confirms that this system is very useful in practice as it can be used as a tool to guide the solution of the Kemeny aggregation problem because it allows to estimate runtimes for larger profiles of rankings and consequently to decide in advance whether the problem is addressable or not. It also provides a parametrization of the runtime for new algorithms based on the characteristics of the profile.

5.5 Identification of the fastest profiles of rankings

It might be also interesting to look only for the fastest profiles, especially when n increases an consequently runtime increases. To do that, using the profiles defined in Sect. 4, the output variable is categorized either as *fast* or *slow* depending on its runtime in relation to the profiles with the same number of alternatives. In this case, the profiles in Q_1 (first quartile) are labeled as *fast* and the remaining profiles as *slow*. The model trained using this dataset with the random forest algorithm obtains an AUC of 0.8173 and a sensitivity of 0.8164.

The 5 most important indices to predict the output using this model are (1) μ_{11}, (2) μ_{14}, (3) μ_8, (4) μ_{10} and (5) μ_{12}. The mean of $\boldsymbol{\beta}$ appears now in the top variables (μ_{14}) as behaves different depending on the number of voters. The number of different margins appears (μ_8) also as one of the important variables, which intuitively has an impact on the number of rankings to be explored, as when the number of repeated indices is high the alternatives add the same amount to the distance to the profile.

Note that μ_{11} is the most important variable for both identifying the most difficult and the simplest profiles (in terms of the the runtime to get Kemeny ranking using Algorithm 1). Let us study more in depth the behavior of this

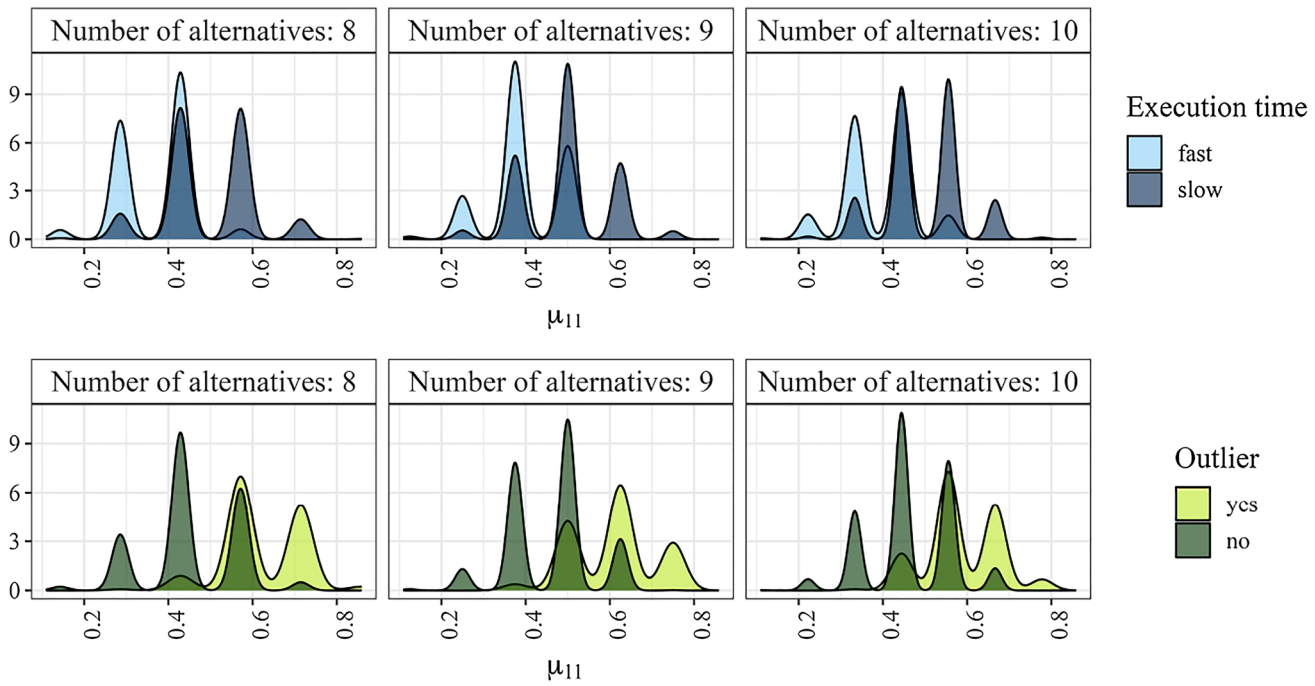

Fig. 2 Distribution of the number of overall preferred alternatives (μ_{11}) by class to predict. The figure above shows the distribution of μ_{11} with respect to the problem studied in 5.4. The figure below shows the distribution of μ_{11} with respect to the problem studied in Sect. 5.3

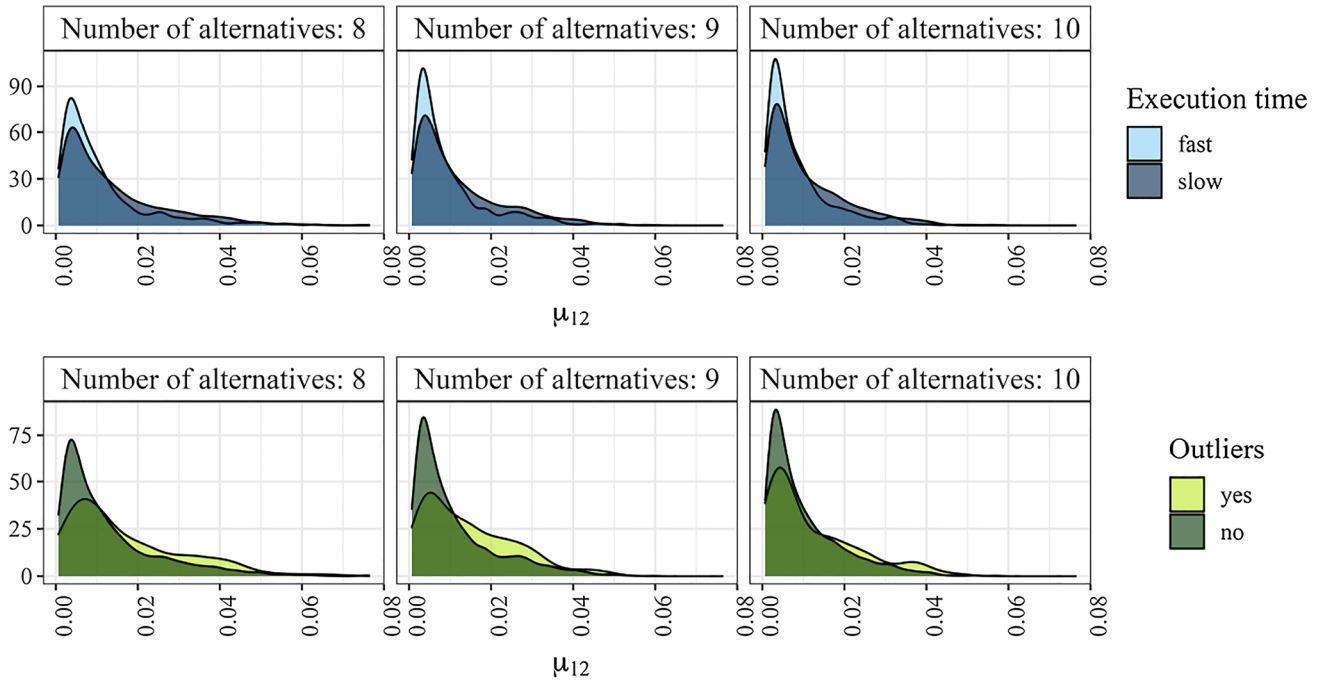

Fig. 3 Distribution of the standard deviation of θ (μ_{12}) for the two classification problems divided by class

important index. Fig. 2 shows the distribution of μ_{11} by class for each problem. For both datasets, the fastest profiles, which are those labeled as *fast* in the first data set and as *no* in the second data set (note that there are not the same set) present a distribution of values to the left for any number of alternatives. In the same line, Fig. 3 shows that the behavior μ_{12} across classes is also different. This trend is repeated for all the other important indices.

6 Discussion and conclusions

6.1 Discussion

The starting point of this work was to develop a method to identify the relevant characteristics of a profile that can be helpful to predict its runtime when Kemeny ranking is computed. These characteristics are expressed in terms of indices, some of them introduced in this work. This proposal is based on exploring a machine learning approach using these characteristics. To test the performance of our proposal, the baseline exact algorithm (Azzini and Munda 2020) is considered.

The results obtained after applying machine learning with the proposed datasets show how some of the indices proposed in this work have an impact on the execution time of the algorithm to find the Kemeny ranking. This means that, the indices collaborate in measuring the difficulty of a profile in terms of execution time.

In fact, according to the experiments shown in Sects. 5.3 to 5.5, it is highlighted that among the top indices relevant for this problem, index μ_{11} is the most important one. It is reasonable as the algorithm is based on the recursive exploration of the alternatives. As it was detailed in Sect. 5.3, when this index is low, greater amounts of potential rankings can be discarded from the exploration. The standard deviation of θ, measured in the μ_{12} index, has also great impact on the execution time. This index is one of the new indices proposed by the authors, and captures how far the index is from what ideally would be the Condorcet ranking if there was not cycles in the preferences of the voters. The fact that the profiles that have larger execution times are further from the standard Condorcet ranking, reflects intuitively more uncertainty among the opinion of the voters, making more difficult to take a decision about the consensus ranking.

Let us remark that the identification of indices is extremely important. Usually the performance of the algorithms that solve the Kemeny problem is tested depending only on the number of alternatives as it is the straightforward factor that influences the factorial growth of the runtime. However, as shown in this work, not all the profiles of rankings with the same number of alternatives and voters are equally difficult to solve attending to their runtime, what is an important finding from our point of view.

What is more important, this work provides a novel methodology to narrow the runtime required to find Kemeny ranking. This approach is based on the some indices that describe the profile and on machine learning. The solution

presented in this paper is not universal for any kind of algorithm in the sense that the indices identified as relevant for predicting runtime are dependent on the algorithm used to obtain Kemeny ranking.

In any case, the approach proposed introduces the characterization of the profiles based on indices as well as the application of machine learning to predict the behavior of the ranking aggregation method and the experiments developed confirm that this methodology is useful to produce reliable predictions for the behavior of the execution times for this complex problem. If the same methodology is applied to predict runtime using a different algorithm (to obtain Kemeny ranking), the proposed procedure provides a different characterization of the profiles. That means that the set of more relevant indices could be different to the obtained in this work as the baseline algorithm is also different. These generalization ability is possible thanks to the large set of indices that are also presented in the paper to represent each profile (many of them introduced by us).

In this work it is also provided a benchmarking algorithm that is publicly available. Note that the main contributions of this paper, namely, the machine learning methodology, the indices used to represent the profiles as well as the construction of bechmarking datasets are independent of the algorithm.

6.2 Conclusion

In this work we study the runtime required for solving the Kemeny problem depending on the characteristics of the profile of rankings by using indices to measure different aspects. Using this characterization based on indices, a dataset is built by obtaining the proposed indices for different profiles of rankings. This dataset is used as input of explainable machine learning models that help in predicting runtime behavior prior to the execution of the algorithm in order to ensure its feasibility. Moreover, the models obtained show that some of the indices proposed by us, such as the overall number of preferred alternatives, can be used as predictors of the runtime required to obtain the Kemeny ranking given a profile of rankings.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Availability of data and material All data generated or analysed during this study are available at http://github.com/noeliarico/consensus_benchmark.

References

Ali A, Meila M (2012) Experiments with Kemeny ranking: what works when? Math Soc Sci 64(1):28–40

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

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

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

Betzler N, Fellows MR, Guo J, Niedermeier R, Rosamond FA (2008) Fixed-parameter algorithms for Kemeny scores, volume 5034 of lecture notes in computer science. Springer, Berlin, pp 60–71

Betzler N, Fellows MR, Guo J, Niedermeier R, Rosamond FA (2009) Fixed-parameter algorithms for Kemeny rankings. Theor Comput Sci 410(45):4554–4570

Borda JC (1781) Mémoire sur les Élections au Scrutin. Histoire de l'Académie Royale des Sciences, Paris

Brandt F, Conitzer V, Endriss U, Lang J, Procaccia AD (eds) (2016) Handbook of computational social choice. Cambridge University Press, Cambridge

Condorcet M (1785) Essai sur l'Application de l'Analyse à la Probabilité des Décisions Rendues à la Pluralité des Voix. De l'Imprimerie Royale, Paris

Fishburn PC (1973) The theory of social choice. Princeton University Press, Princeton

Greenwell BM, Boehmke BC (2020) Variable importance plots-an introduction to the VIP package. R J 12(1):343–366

Hemaspaandra E, Spakowski H, Vogel J (2005) The complexity of Kemeny elections. Theor Comput Sci 349(3):382–391

Kemeny JG (1959) Mathematics without numbers. Daedalus 88(4):577–591

Kendall M (1938) A new measure of rank correlation. Biometrika 30(1/2):81–93

Kubat M (2017) An introduction to machine learning, 2nd edn. Springer, Berlin

Lunardon N, Menardi G, Torelli N (2014) ROSE: a package for binary imbalanced learning. R J 6(1):82–92

Oliveira SE, Diniz V, Lacerda A, Merschmanm L, Pappa GL (2020) Is rank aggregation effective in recommender systems? An experimental analysis. ACM Trans Intell Syst Technol 11(2):1–26

Pérez-Fernández R, Rademaker M, Alonso P, Díaz I, Montes S, De Baets B (2016) Representations of votes facilitating monotonicity-based ranking rules: from votrix to votex. Int J Approx Reason 73:87–107

Rico N, Vela CR, Díaz I (2021a) An analysis of the indexes measuring the agreement of a profile of rankings. Accepted in XIX CAEPIA

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

Schulze M (2011) A new monotonic, clone-independent, reversal symmetric, and Condorcet-consistent single-winner election method. Soc Choice Welf 36(2):267–303

Young HP, Levenglick A (1978) A consistent extension of Condorcet's election principle. SIAM J Appl Math 35(2):285–300