Computers and Operations Research 98 (2018) 127–136

Contents lists available at ScienceDirect

Computers and Operations Research

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

Mathematical programming formulations for the efficient solution of the *k*-sum approval voting problem

Diego Ponce[a], Justo Puerto[a], Federica Ricca[b], Andrea Scozzari[c,*]

[a] IMUS and Departamento de Estadística e Investigación Operativa, Universidad de Sevilla, Spain
[b] Università di Roma, La Sapienza, Italy
[c] Università degli Studi Niccolò Cusano, Roma, Italy

ARTICLE INFO

Article history:
Received 24 July 2017
Revised 27 March 2018
Accepted 14 May 2018
Available online 24 May 2018

Keywords:
Approval voting
Ordered Weighted Averaging (OWA)
k-sum optimization problems

ABSTRACT

In this paper we address the problem of electing a committee among a set of m candidates on the basis of the preferences of a set of n voters. We consider the approval voting method in which each voter can approve as many candidates as he likes by expressing a preference profile (boolean m-vector). In order to elect a committee, a voting rule must be established to 'transform' the n voters' profiles into a winning committee. The problem is widely studied in voting theory; for a variety of voting rules the problem was shown to be computationally difficult and approximation algorithms and heuristic techniques were proposed in the literature. In this paper we follow an Ordered Weighted Averaging approach and study the k-sum approval voting (optimization) problem in the general case $1 \leq k < n$. For this problem, we provide different mathematical programming formulations that allow us to solve it in an exact solution framework. We provide computational results showing that our approach is efficient for medium size test problems (n up to 200, m up to 60), since in all tested cases it was able to find the exact optimal solution in very short computational times.

1. Introduction

A typical problem in collective decision making is to select one (or more) winners among a set of m candidates on the basis of the preferences of a set of n voters. This situation arises in many different real-life contexts, as in sport competitions to select the set of winners, or in political elections, and, more in general, whenever a committee must be formed from a larger set of candidates (for example, in companies or universities). For $m > 1$ the problem is a *multi-winner* one and the ballot gives the possibility to each voter to express his preference for each single candidate by *approving* or not his nomination. This means that voters can approve as many candidates as they like by expressing their preference profile (*approval balloting*). Approval voting is a well-known method used for this kind of multi-winner elections. The method, introduced by Brams and Fishburn in 1978 Brams and Fishburn (1978), was widely studied in the literature on voting theory (see Brams and Fishburn, 2005; Brams and Fishburn, 2007; Laslier and Sanver, 2010; Menezes et al., 2016; Rapoport and Felsenthal, 1990; Taylor and Pacelli, 2008 and the references therein).

Consider a set N of n voters and a set A of m candidates. For each i, $P_i = (p_{i1}, \ldots, p_{ij}, \ldots, p_{im})$ denotes the preference profile of voter i which corresponds to a boolean m-vector whose generic element p_{ij} is equal to 1 if candidate j is approved by i and equal to 0 otherwise. Therefore, in the problem input we have N, A and a set P of preference profiles of the voters, so that a generic instance is denoted by (N, A, P). The problem is to find in A the "best" subset of candidates (*winning* or *elected* committee), according to a certain voting rule (criterion). A committee is represented by a boolean m-vector $x = (x_1, \ldots, x_j, \ldots, x_m)$, in which $x_j = 1$ means that candidate j is in the committee, and $x_j = 0$ otherwise.

When we have a single-winner election, the typical voting rule is the one that elects the most approved candidate, i.e., the one who received the largest number of votes (with a tie-breaking rule if needed).

For multi-winner elections, several voting rules have been proposed for approval voting (Kilgour, 2010). For a majority of the voting rules computing a winning committee is a difficult problem (Aziz et al., 2015; Fishburn and Pekec, 2004). Among the many, there is a class of rules known as *centralization procedures* that was widely studied in the literature. Two rules in this class were mainly analyzed, namely, one based on the *minisum* criterion and one on the *minimax* criterion. According to the first criterion, the winning committee is the one that minimizes the sum

* Corresponding author.
E-mail addresses: dponce@us.es (D. Ponce), puerto@us.es (J. Puerto), federica.ricca@uniroma1.it (F. Ricca), andrea.scozzari@unicusano.it (A. Scozzari).

https://doi.org/10.1016/j.cor.2018.05.014
0305-0548/

of the n Hamming distances to the preferences profiles of the n voters. The second criterion selects the committee that minimizes the maximum of its Hamming distances to the voters' profiles. In Amanatidis et al. (2015), a new family of rules has been proposed to generalize minisum and minimax which make use of *Ordered Weighted Averaging* operators (OWA) introduced by Yager (1988) (see also Barrot et al., 2017). In this setting, a vector of n weights $W = (w_1, \ldots, w_n)$ is fixed; then the n distances between voters' profiles and the decision vector (committee) are ordered from largest to smallest and they are weighted with the corresponding weight in W. Clearly, when $W = (1, 0, \ldots, 0)$ we have the minimax criterion, while for $W = (\frac{1}{n}, \ldots, \frac{1}{n})$ we have the minisum criterion. We observe that in Yager (1988) the weights are normalized w.r.t. n. Since we do not normalize, the minisum criterion in our case is given by $W = (1, 1, \ldots, 1)$. Many other criteria can be defined in this way by tuning the weights in W according to the specific goal of the application. An interesting class of problems arises when vector W has only 0/1 values and more than one weight is equal to 1. Suppose to have k elements equal to 1; when they are in the first k positions of the vector of weights they refer to the k largest distances, thus providing what is known in the literature as the k-sum approach already applied to many other combinatorial problems (Nickel and Puerto, 2006). A similar problem arises when we have elements equal to 1 in the last h positions of the weighting vector (h smallest distances). Both problems have meaningful applications in approval voting.

In this paper we study the problem of selecting a committee by applying approval voting and basing on a k-sum objective function (k-sum approval voting problems). In Amanatidis et al. (2015) it is proved that for $1 \leq k < n$ the problem is NP-hard and, therefore, an approximation algorithm is provided to get feasible solutions with guaranteed bounds. On the other hand, the same authors provide polynomial time exact algorithms for some families of weighting vectors that consider the h smallest distances (h fixed). In the present paper we study these kind of problems under a mathematical programming viewpoint, providing different exact formulations for the k-sum approval voting problem, with $1 \leq k < n$. We then exploit these formulations to develop exact solution procedures that may be used to solve medium size problems at optimality, or to efficiently find a sub-optimal solution when the size of the problem is too large. In approval voting problems we consider as small an instance with $n < 100$ and $m < 30$. A medium size instance has $n \in [100, 200]$ and $m \in [30, 60]$. Medium size problems fit committee elections in universities or companies. On the other hand, large size problems arise in political elections where typically $n \gg 200$. To develop such formulations we rely on the general approach for solving k-sum optimization problems provided in Ogryczak and Tamir (2003) and in Puerto et al. (2016), as well as, on theoretical results in Blanco et al. (2014). We experimentally study the solution of our k-sum approval voting problem by using the above formulations in a Branch & Bound framework. All formulations were tested on a variety of medium size randomly generated test problems, in all cases providing the exact optimum in very short times. In view of this, our formulations also provide an efficient tool to certify optimality of a solution.

We apply the mathematical programming approach also when the h smallest distances are considered in the objective function (anti-h-sum approval voting problem). We formulate this problem as a polynomial sequence of linear programs, thus providing a formal proof that it can be solved in polynomial time as already established in Amanatidis et al. (2015).

The paper is organized as follows. Section 2 formally defines the problem and sets the notation. Section 3 presents our mathematical programming formulations for the k-sum approval voting problem minimizing the sum of the k largest Hamming distances. We have developed two different types of formulations. The first ones based on an exponential number of constraints that can be separated efficiently (see Section 3.1) and the second ones based on compact representations of k-sums (see Section 3.2). In Section 4 we describe how all the above mentioned formulations can be strengthened with variable fixing and valid inequalities. The computational experiments are reported in Section 5, where we compare the performance of the formulations on two different sets of instances. Our results show that these formulations are able to solve the problem for instances of medium and large sizes. Next, Section 6 outlines how to proceed for the anti-h-sum approval voting problem which consists of minimizing the sum of the h smallest distances with respect to a given profile Amanatidis et al. (2015). Finally, Section 7 contains our concluding remarks.

2. Problem definition and basic results

Consider an instance of the k-sum approval voting problem (N, A, P). For a given committee x (i.e., a boolean vector x of length m) we compute the Hamming distance between x and each voter profile P_i, $i = 1, \ldots, n$, thus obtaining the Hamming distance $d_i(x)$ for each voter i. Following the OWA approach, we consider a family of functions, parameterized by a vector of length n that maps a vector of distances $(d_1(x), \ldots, d_n(x))$ to an aggregated function $D(x)$ (OWA score). The k-sum approval voting problem can be stated as follows: select a committee x minimizing the OWA score of Hamming distances $D(x)$. In this paper we study two families of k-sum approval voting problems. The first computes the OWA score using the following vector of weights:

$$W(k) = (1, \ldots, 1, 0, \ldots, 0),$$

where k refers to the number of ones in the first k positions of vector W (electing a committee that minimizes the sum of the k largest Hamming distances).

The second family uses the following weighting vector:

$$M(n-h) = (0, \ldots, 0, 1, \ldots, 1),$$

where h weights equal to 1 are in the last h positions (electing a committee that minimizes the sum of the h smallest Hamming distances). Clearly, the cases $k = h = n$ are the same, and, in this case, we have the same OWA problem, that, in fact, corresponds to the minisum problem.

In Brams et al. (2007) (see Proposition 4) the authors account for why the minisum problem is polynomially solvable, but they do not provide a formal proof. The key idea is that the minisum winning committee (also under a cardinality constraint on the size of the committee fixed to C), corresponds to a set of candidates receiving the most votes. In the following, we illustrate how the result in Brams et al. (2007) can be formally obtained by exploiting Linear Programming (LP). Denote by γ_j the number of votes for candidate j, $j = 1, \ldots, m$, we have

$$\gamma_j = \sum_{i=1}^{n} p_{ij}.$$

Consider the case when the size of the committee is not given. A valid formulation can be obtained basing on the following observation (see Brams et al., 2007): when $k = n$, all voters' Hamming distances $(d_1(x), \ldots, d_n(x))$ are considered in the objective function, so that a candidate j is elected if the number of votes for him γ_j is greater than $n - \gamma_j$. This leads to the following model

$$\min \sum_{j=1}^{m} (n - \gamma_j) x_j + \sum_{j=1}^{m} \gamma_j (1 - x_j)$$

$$\text{s.t. } x \in \{0, 1\}^m.$$

Since the objective function is linear, the optimal solution is attained at some vertex of the m-dimensional hypercube. Then, we can relax the binary variables of the above problem obtaining the LP model

$$\min \sum_{j=1}^{m}(n-\gamma_j)x_j + \sum_{j=1}^{m}\gamma_j(1-x_j)$$
$$\text{s.t. } 0 \le x_j \le 1, \quad j = 1,\dots,n.$$

As in Brams et al. (2007), we can also consider a constraint on the size C of the committee, obtaining the following model

$$\begin{aligned}\min \ & \sum_{j=1}^{m}(n-\gamma_j)x_j + \sum_{j=1}^{m}\gamma_j(1-x_j)\\ \text{s.t.} \ & \sum_{j=1}^{m} x_j = C\\ & x \in \{0,1\}^m. \end{aligned} \tag{1}$$

Since the constraint matrix is Totally Unimodular (TU), the problem can be still solved by Linear Programming techniques, after relaxing the binary constraints on the variables x (see Nemhauser and Wolsey, 1988). This definitely shows that the minisum case is polynomially solvable.

When $k < n$ the k-sum approval voting problem is NP-hard (see Amanatidis et al., 2015; Frances and Litman, 1997). This justifies the idea of studying efficient solution methods for the general problem resorting to heuristic approaches, or approximation algorithms (Byrka and Sornat, 2014; Caragiannis et al., 2010; LeGrand et al., 2007; Li et al., 2002). In the following sections we present a number of exact mathematical programming formulations of the general k-sum approval voting problem with $1 \le k < n$ that can be efficiently solved in a Branch & Bound framework enriched by the use of valid inequalities.

A similar problem arises when we consider the weighting vector $M(n-h)$ with $1 \le h < n$. The computational complexity of this problem was established in Amanatidis et al. (2015) when h is not part of the input of the problem. Even if it is shown that the problem is polynomially solvable in this case, to the best of our knowledge, the computational complexity is still open when the problem is formulated with a general h. We discuss this problem in the final sections of the paper.

3. Mathematical programming formulations for the approval voting problem

Consider now $W(k)$ with $1 \le k < n$. Let x be a committee, the Hamming distance, $d_i(x)$, between the profile P_i of voter i and x is given by

$$d_i(x) = \sum_{j=1}^{m} |p_{ij} - x_j|.$$

Since both P_i, $i = 1,\dots,n$, and x are boolean vectors, we can exploit the fact that the Hamming distance between two boolean vectors corresponds to the vector of the *Exclusive-or* between the vector elements (Givant and Halmos, 2009). Thus the Hamming distance $d_i(x)$ can be rewritten as follows

$$\begin{aligned} d_i(x) &= \sum_{j=1}^{m} z_{ij}\\ z_{ij} &\ge x_j - p_{ij}x_j \quad j = 1,\dots,m\\ z_{ij} &\ge p_{ij} - p_{ij}x_j \quad j = 1,\dots,m. \end{aligned} \tag{2}$$

We can also replace the two inequalities above by the equivalent representation: $z_{ij} \ge x_j(1-p_{ij}) + p_{ij}(1-x_j)$. This gives rise to:

$$d_i(x) = \sum_{j=1}^{m} z_{ij} \tag{3}$$

$$z_{ij} \ge x_j(1-p_{ij}) + p_{ij}(1-x_j), \quad j = 1,\dots,m. \tag{4}$$

Let $\sigma_x : \{1,\dots,n\} \to \{1,\dots,n\}$ be an ordering function that, for a given x, provides a permutation of the voters' indices such that $d_{\sigma_x(1)}(x) \ge d_{\sigma_x(2)}(x) \ge \dots \ge d_{\sigma_x(n)}(x)$. For the given permutation the problem of electing a committee that minimizes the sum of the k largest distances can be formulated as follows:

$$\begin{aligned}\min \ & \sum_{h=1}^{k} d_{\sigma_x(h)}(x)\\ & x \in \{0,1\}^m. \end{aligned} \tag{5}$$

Following the approach in Blanco et al. (2014), the above problem can be restated as:

$$\min_{x\in\{0,1\}^m} \left(\max\left\{d_S(x) \mid S \subset \{1,\dots,n\}, |S| = k\right\}\right), \tag{6}$$

where $d_S(x) = \sum_{i\in S}\sum_{j=1}^{m} |p_{ij} - x_j|$, is the Hamming distance of the set of voters in S to the committee represented by x.

In general k-sum problems, the expression of the contribution of a subset of voters to the election of a candidate can be also written by means of the γ_j values, namely the number of voters approving a given candidate j. For this purpose, let us introduce some necessary notation. More formally, let S be a set of voters such that $|S| = k$. We define $\gamma_j(S) = \sum_{i\in S} p_{ij}$, as the number of votes of candidate j by the voters in S. For a given x and S such that $|S| = k$, we can express:

$$d_S(x) = \sum_{j=1}^{m} \max\{\gamma_j(S)(1-x_j), (k-\gamma_j(S))x_j\} \tag{7}$$

$$= \sum_{j=1}^{m} \gamma_j(S)(1-x_j) + \sum_{j=1}^{m}(k-\gamma_j(S))x_j, \tag{8}$$

i.e., the Hamming score of the k voters in S computed w.r.t. a given solution x. Notice that by means of expressions (7) and (8) the score is well calculated even when the solution is not optimal. Based on these expressions, in the following sections we obtain alternative valid formulations of the k-sum approval voting problem that we then test experimentally in Section 5.

3.1. Valid formulations based on subsets of size k

In this section we propose a first valid formulation for our k-sum approval voting problem which is based on expression (7). We formulate it in the following theorem where, for the sake of simplicity, we avoid specifying $S \subset \{1,\dots,n\}$ when not necessary.

Theorem 1. *An optimal solution of the k-sum approval voting problem can be obtained solving the following integer programming problem.*

$$\min \nu \tag{9}$$

$$\text{s.t. } z_{ij} \ge p_{ij}(1-x_j) \quad \forall i, j \tag{10}$$

$$z_{ij} \ge (1-p_{ij})x_j \quad \forall i, j \tag{11}$$

$$\sum_{j=1}^{m}\sum_{i\in S} z_{ij} \le \nu \quad \forall S : |S| = k \tag{12}$$

$$x \in \{0,1\}^m. \tag{13}$$

Proof. Consider formulation (6) and formula (7), the problem can be written as

$$\min_{x\in\{0,1\}^m}\left(\max\left\{\sum_{j=1}^{m}\max\{(k-\gamma_j(S))x_j,\gamma_j(S)(1-x_j)\}|S\subset\{1,\ldots,n\},\ |S|=k\right\}\right).$$

Defining variables z_{Sj} for all $S\subset\{1,\ldots,n\}$, $|S|=k$, and all j, this is equivalent to

$$\min \nu \tag{14}$$

$$\text{s.t. } z_{Sj} \geq \gamma_j(S)(1-x_j) \qquad \forall j,\ \forall S:|S|=k \tag{15}$$

$$z_{Sj} \geq (k-\gamma_j(S))x_j \qquad \forall j,\ \forall S:|S|=k \tag{16}$$

$$\sum_{j=1}^{m} z_{Sj} \leq \nu \qquad \forall S:|S|=k \tag{17}$$

$$x \in \{0,1\}^m. \tag{18}$$

Next, we can define variables z_{ij} for all $i=1,\ldots,n$ and for all $j=1,\ldots,m$ and disaggregate each variable $z_{Sj}=\sum_{i\in S} z_{ij}$, which results in

$$\min \nu \tag{19}$$

$$\text{s.t. } \sum_{i\in S} z_{ij} \geq \gamma_j(S)(1-x_j) \qquad \forall j,\ \forall S:|S|=k \tag{20}$$

$$\sum_{i\in S} z_{ij} \geq (k-\gamma_j(S))x_j \qquad \forall j,\ \forall S:|S|=k \tag{21}$$

$$\sum_{j=1}^{m}\sum_{i\in S} z_{ij} \leq \nu \qquad \forall S:|S|=k \tag{22}$$

$$x \in \{0,1\}^m. \tag{23}$$

We observe that constraints (20) and (21) for all j and S such that $|S|=k$, can be replaced by the following disaggregated version

$$\begin{aligned}
&\min \nu\\
&\text{s.t. } z_{ij} \geq p_{ij}(1-x_j) \quad \forall i,j\\
&\quad z_{ij} \geq (1-p_{ij})x_j \quad \forall i,j\\
&\quad \sum_{j=1}^{m}\sum_{i\in S} z_{ij} \leq \nu \quad \forall S:|S|=k\\
&\quad x\in\{0,1\}^m.
\end{aligned}$$

This concludes the proof. □

Example 1. We illustrate the above approach reformulating the minimax approval voting problem ($k=1$) within this general framework.

$$\begin{aligned}
&\min \quad \nu\\
&z_{ij} \geq p_{ij}(1-x_j)\ \forall i,j\\
&z_{ij} \geq (1-p_{ij})x_j\ \forall i,j\\
&\sum_{j=1}^{m} z_{ij} \leq \nu\ \forall i\\
&x\in\{0,1\}^m.
\end{aligned} \tag{24}$$

In the following, we develop a second valid formulation for the general k-sum approval voting problem applying (6) but using (8) to represent the Hamming distance instead of (7).

Theorem 2. *The following formulation provides a valid representation of the k-sum approval voting problem.*

$$\min \nu \tag{25}$$

$$\text{s.t. } \sum_{j=1}^{m}(k-\gamma_j(S))x_j + \sum_{j=1}^{m}\gamma_j(S)(1-x_j) \leq \nu \quad \forall S:|S|=k \tag{26}$$

$$x \in \{0,1\}^m. \tag{27}$$

Proof. Applying in formula (6) the representation (8) instead of (7), the proof follows similarly to that of Theorem 1. □

Since both formulations (9)–(13) and (25)–(27) have an exponential number of constraints, we propose here two different approaches to solve these two models.

A first approach is based on formulation (9)–(13). Let us assume that we embed that formulation in a Branch and Bound scheme and let $(\hat{x},\hat{z},\hat{v})$ be the current solution in a node of this Branch & Bound tree.

Procedure for (9)–(13)

- Compute $\hat{r}_i := \sum_{j=1}^{m}\hat{z}_{ij}$ for every i and choose the k largest. Determine $\hat{S}$ according to the k largest $\hat{r}_i$ for $i\in\{1,\ldots,n\}$.
- Check if $\sum_{i\in\hat{S}}\hat{r}_i > \hat{v}$. In the affirmative case, we need to add the following constraint related to such $\hat{S}$

$$\sum_{i\in\hat{S}}\sum_{j=1}^{m} z_{ij} \leq \nu. \tag{28}$$

Otherwise, i.e., when the answer to the test is no, the current solution is feasible in the current node. Therefore, a valid description of the problem was already available and no more inequalities have to be added.

A similar scheme can be applied to Problem (25)–(27). Let $(\hat{x},\hat{v})$ be a given feasible solution in a node of its Branch & Bound tree.

Procedure for (25)–(27)

- Compute $\hat{r}_i := \sum_{j=1}^{m}|\hat{x}_j - p_{ij}|$, for all $i=1,\ldots,n$. Determine $\hat{S}$ according to the k largest values of $\hat{r}_i$ for $i\in\{1,\ldots,n\}$.
- Check whether $\sum_{i\in\hat{S}}\hat{r}_i > \hat{v}$. In the affirmative case we need to add the following inequality which is a valid cut that separates $(\hat{x},\hat{v})$

$$\sum_{j=1}^{m}\gamma_j(\hat{S})(1-x_j) + \sum_{j=1}^{m}(k-\gamma_j(\hat{S}))x_j \leq \nu.$$

Remark 1. Formulation (25)–(27) is at least as good as formulation (9)–(13).

The above result can be explained as follows. Let $P_{(25)-(27)}$ and $P_{(9)-(13)}$ be the polyhedra defining the feasible domains of the continuous relaxation of formulations (25)–(27) and (9)–(13), respectively. Consider a feasible solution (possibly fractional) $(x,v,z)\in P_{(9)-(13)}$. It follows that its projection onto (x,v) belongs to $P_{(25)-(27)}$, and the result follows.

The above result, together with the fact that formulation (25)–(27) has a smaller number of constraints and variables than formulation (9)–(13), justifies that in our computational experiments (see Section 5) we only report results based on formulation (25)–(27) since its performance is superior to the one of (9)–(13).

3.2. Valid formulations based on Hamming distance among profiles

In this section, we derive alternative valid formulations for the k-sum approval voting problem that do not make explicit use of all the possible subsets of $\{1,\dots,n\}$ of cardinality k. For an arbitrary subset S of $\{1,\dots,n\}$, we consider the sum of the Hamming distances of all P_i, $i\in S$, to x as follows

$$d_S(x)=\sum_{i\in S}\sum_{j=1}^{m}|x_j-p_{ij}|.$$

For a given k, with $1\le k<n$, the problem of electing a committee that minimizes the sum of the k largest Hamming distances can be formulated as a Mixed Integer Linear Programming (MILP) problem, provided that for any given x a permutation σ_x such that $d_{\sigma_x(i)}(x)\ge d_{\sigma_x(i+1)}(x),\ i=1,\dots,n-1,$ is fixed.

$$\min\sum_{h=1}^{k}d_{\sigma_x(h)}(x) \tag{29}$$

$$z_{ij}\ge x_j(1-p_{ij})+p_{ij}(1-x_j)\quad i=1,\dots,n,\ j=1,\dots,m \tag{30}$$

$$d_i(x)\ge\sum_{j=1}^{m}z_{ij}\quad i=1,\dots,n \tag{31}$$

$$x_j\in\{0,1\}\quad j=1,\dots,m. \tag{32}$$

Problem (29)–(32) has m binary variables, nm continuous variables, and $O(nm)$ linear constraints. If there is no confusion, in the following, we simply write d_i in place of $d_i(x)$.

The above formulation is correct but it is not operational, since it depends on the valid permutation function $\sigma_x(\cdot)$ that sorts the distances in a non-increasing order. However, it is still possible to derive alternative valid formulations that do not make explicit use of that permutation (see Ogryczak and Tamir, 2003; Puerto et al., 2016). Indeed, let us consider a new variable $t\ge 0$ and a set of n variables v_i, $i=1,\dots,n$.

Theorem 3. *The following is a valid formulation for the general k-sum approval voting problem.*

$$\min kt+\sum_{i=1}^{n}v_i \tag{33}$$

$$\text{s.t. } v_i\ge d_i-t\qquad i=1,\dots,n \tag{34}$$

$$d_i\ge\sum_{j=1}^{m}z_{ij}\qquad i=1,\dots,n \tag{35}$$

$$z_{ij}\ge x_j(1-p_{ij})+p_{ij}(1-x_j)\qquad i=1,\dots,n,\ j=1,\dots,m \tag{36}$$

$$x_j\in\{0,1\}\qquad j=1,\dots,m \tag{37}$$

$$t\ge 0,\ v_i\ge 0\qquad i=1,\dots,n. \tag{38}$$

Proof. Consider the general formulation (6). Following the proof in Puerto et al. (2016), the inner maximum in problem (6) is equivalent to the following

$$\begin{aligned}\max\ &\sum_{i=1}^{n}d_iq_i\\ &\sum_{i=1}^{n}q_i=k\\ &q_i\in\{0,1\}\quad i=1,\dots,n.\end{aligned} \tag{39}$$

The above constraint matrix is TU and thus the integrality constraints on the variables q_i, $i=1,\dots,n$, in problem (39) can be relaxed to $0\le q_i\le 1$, $i=1,\dots,n$, and the resulting problem has the following exact dual:

$$\begin{aligned}\min\ &kt+\sum_{i=1}^{n}v_i\\ \text{s.t. }&v_i\ge d_i-t\quad i=1,\dots,n\\ &v_i\ge 0\quad i=1,\dots,n.\end{aligned} \tag{40}$$

Notice that, since d_i are distances, the coefficients in the objective function of (39) are non negative and, w.l.o.g., we can set the variable t as $t\ge 0$. To complete the proof, it suffices to add to the above dual problem the constraints (30)–(32). □

When a constraint on the number C of candidates to be elected must be also considered, we can add it to the above program by condition

$$\sum_{j=1}^{m}x_j\le C.$$

An alternative valid formulation for the general k-sum approval voting problem can be provided by exploiting the one proposed in Blanco et al. (2014), as stated in the following theorem.

Theorem 4. *The following is a valid formulation for the general k-sum approval voting problem.*

$$\min\sum_{i=1}^{n}u_i+\sum_{h=1}^{k}v_h \tag{41}$$

$$\text{s.t. } u_i+v_h\ge d_i\qquad i=1,\dots,n,\ h=1,\dots,k \tag{42}$$

$$d_i\ge\sum_{j=1}^{m}z_{ij}\qquad i=1,\dots,n \tag{43}$$

$$z_{ij}\ge x_j(1-p_{ij})+p_{ij}(1-x_j)\qquad i=1,\dots,n,\ j=1,\dots,m \tag{44}$$

$$x_j\in\{0,1\}\qquad j=1,\dots,m \tag{45}$$

$$u_i\ge 0\qquad i=1,\dots,n \tag{46}$$

$$v_h\ge 0\qquad h=1,\dots,k. \tag{47}$$

Proof. Consider the general formulation (6). We introduce the following binary variables y_{ih}, $i=1,\dots,n$ and $h=1,\dots,k$, such that, given x, $y_{ih}=1$ if the distance d_i of voter i is in position $h\le k$ in the non-increasing ordering, and $y_{ih}=0$ otherwise. Following the proof in Blanco et al. (2014), for a given vector x, the sum of the k largest distances can be written

$$\begin{aligned}\sum_{h=1}^{k} d_{\sigma_x(h)}(x) = \max \quad & \sum_{i=1}^{n}\sum_{h=1}^{k} d_i y_{ih} \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_{ih} \le 1 \qquad h = 1,\dots,k \\ & \sum_{h=1}^{k} y_{ih} \le 1 \qquad i = 1,\dots,n \\ & y_{ih} \in \{0,1\} \qquad i = 1,\dots,n,\ h = 1,\dots,k. \end{aligned} \tag{48}$$

In fact, problem (48) is a transportation problem, so that we can relax the binary variables $0 \le y_{ih} \le 1$. Taking the dual of (48) we obtain

$$\begin{aligned}\sum_{h=1}^{k} d_{\sigma_x(h)}(x) = \min \quad & \sum_{i=1}^{n} u_i + \sum_{h=1}^{k} v_h \\ \text{s.t.} \quad & u_i + v_h \ge d_i \qquad i = 1,\dots,n,\ h = 1,\dots,k \\ & u_i,\ v_h \ge 0 \qquad i = 1,\dots,n,\ h = 1,\dots,k. \end{aligned} \tag{49}$$

To complete the proof, it suffices to add to the above dual problem the constraints (30)–(32). □

Remark 2. The equivalence between formulations (33)–(38) and (41)–(47), in Theorems 3 and 4, implies that the formulation in Theorem 4 admits, at least, an optimal solution in the form $v_h^* = t^*$ for all $h = 1,\dots,k$ and $u_i^* = \max\{0, t^*\}$, for all $i = 1,\dots,n$ that is also optimal for the formulation of Theorem 3.

Proposition 1. *The formulations (25)–(27), (33)–(38) and (41)–(47) produce the same LP bound.*

Proof. Let us consider a generic $\hat{x} \in [0,1]^m$. From our discussion above, it is clear that, fixing $\hat{x}$, the objective function value of all problems (25)–(27), (33)–(38) and (41)–(47) equals $\sum_{h=1}^{k} d_{\sigma_{\hat{x}(h)}}(\hat{x})$, where $d_{\sigma_{\hat{x}(1)}}(\hat{x}) \ge \dots \ge d_{\sigma_{\hat{x}(n)}}(\hat{x})$. Hence, the three problems return the same objective function value for each feasible solution of the continuous polytope and therefore this proves the claim. □

4. Strengthening the formulations

The above MIP formulations are exact but still one can observe that they have some GAP at the root node of the Branch & Bound tree (see Section 5) although this gap is always rather small. The goal of this section is to develop some preprocessing strategies and valid inequalities that allow to improve the polyhedral description of the different formulations, and get a better bound for this GAP with a consequent gain in computational times. In the rest of the paper we consider the unconstrained version of the k-sum approval voting problem, i.e., the size of the committee is not fixed.

First of all, we advance an easy preprocessing that allows fixing some variables either to zero or to one before the global search starts. The rationale behind this is the following. For a candidate j to be member of at least one committee it is required that, at least for a subset S of size $k < n$, there is a majority of voters that approves him, that is, $\gamma_j(S) \ge \lfloor k/2 \rfloor$. Therefore, if the total number of voters preferring candidate j, γ_j, satisfies $\gamma_j \ge n - \lfloor k/2 \rfloor$ this candidate will be always included in any committee and thus we can set $x_j = 1$. This justifies (50). On the other hand, if candidate j is only preferred by less than $\lfloor k/2 \rfloor$ voters, he will be never included in a k-sum committee and thus $x_j = 0$. This justifies (51)

$$x_j = 1 \qquad \text{if } \gamma_j \ge n - \lfloor k/2 \rfloor \tag{50}$$

$$x_j = 0 \qquad \text{if } \gamma_j \le \lfloor k/2 \rfloor. \tag{51}$$

Note that this preprocessing is more interesting for large values of k's.

In the following, we also develop a procedure for the efficient solution of the $\hat{k}$-sum approval voting problem for $\hat{k}$ fixed that works in an iterative fashion starting from $k = n$ and solving a k-sum approval voting problem for all $k = n,\dots,\hat{k}$. This procedure is based on valid inequalities involving optimal objective function values of the k-sum and the $(k+1)$-sum approval voting problems for any k.

First, we analyze whether inequalities (20)–(22) can be adapted to the formulations described in Section 3.2, as valid cuts. Clearly, they are valid inequalities but, as we will see, they do not improve such formulations.

Consider, first, formulation (33)–(38). Note that the cuts in (20)–(22) consist in aggregated forms of constraints (36), (34) and (35), respectively.

$$z_{ij} \ge p_{ij}(1 - x_j) \forall i, j \Rightarrow \sum_{i \in S} z_{ij} \ge \gamma_j(S)(1 - x_j) \quad \forall j$$

$$z_{ij} \ge (1 - p_{ij}) x_j \forall i, j \Rightarrow \sum_{i \in S} z_{ij} \ge (k - \gamma_j(S)) x_j \quad \forall j$$

Hence, the use of them as valid inequalities is not useful. Furthermore, constraint $\sum_{j=1}^{m} \sum_{i \in S} z_{ij} \le kt + \sum_{i \in S} v_i$ can be obtained by means of a natural aggregation of (34) and (35).

In light of the above results, we develop valid inequalities based on solutions of the k-sum approval voting problem for different k values to be used in a strategy that solves problems for different k consecutively.

In order to present the result some additional notation is required. For a given k, let us denote by $z(k)$ and $x(k)$ the optimal objective function value and an optimal solution of the k-sum approval voting problem, respectively.

Proposition 2. *For a given instance (N, A, P) of the k-sum approval voting problem the following inequality holds*

$$z(k) \ge \frac{k}{k+1} z(k+1).$$

Proof. By definition, $z(k)$ is the sum of the k largest Hamming distances of the voters' profiles with respect to $x(k)$. It means that distance in position $k+1$, $d_{\sigma_{x(k)}(k+1)}(x(k))$, satisfies

$$0 \le d_{\sigma_{x(k)}(k+1)}(x(k)) \le \frac{z(k)}{k}.$$

Thus, we can conclude that $x(k)$ is a feasible solution for the $(k+1)$-sum problem and moreover, there exists an upper bound for $z(k+1)$ given by

$$z(k+1) \le z(k) + \frac{z(k)}{k}. \tag{52}$$

The above expression gives a lower bound for $z(k)$, provided that $z(k+1)$ is known

$$z(k) \ge \frac{k}{k+1} z(k+1). \tag{53}$$

□

Since, for a given (N, A, P) we can solve the different k-sum voting problems in any order, after the above result, it is advisable to do it following the non-increasing sequence $k = n,\dots,1$. Indeed, as shown in Section 2, solving the problem for $k = n$ (i.e., the minisum problem) is polynomial. Once this solution and objective function value are found, they can be used to improve the solution for $k = n - 1$ and so on.

We now illustrate an iterative scheme for efficiently solving the k-sum approval voting problem following the strategies illustrated above in this section. This approach has been effectively used in our computational experiments.

Table 1
Summary for $n = 50$, uniform data.

Form.	Time (s)		GAP (%)		Nodes		%Solved root	%Fixed
	Avg	Max	Avg	Max	Avg	Max		
(25)–(27)	65.63	4396.86	0.22	2.63	7388.30	1,092,091	15.89	13.92
(33)–(38)	0.13	2.24	0.22	2.63	276.67	23,170	62.74	13.92
(41)–(47)	0.58	19.99	0.22	2.63	1128.03	158,152	54.17	13.92

Table 2
Summary for $n = 50$, biased data.

Form.	Time (s.)		GAP (%)		Nodes		%Solved root	%Fixed
	Avg	Max	Avg	Max	Avg	Max		
(25)–(27)	12.34	258.85	0.35	3.13	973.61	52,028	17.83	29.42
(33)–(38)	0.06	0.40	0.35	3.13	15.76	959	68.91	29.42
(41)–(47)	0.17	1.23	0.35	3.13	71.33	10,914	54.86	29.42

1. Solve the problem for $k = n$, i.e., the minisum problem. Its optimal solution, $x(n)$, is easily seen to be

$$x_j = 1 \qquad \text{if } \gamma_j \geq n - \lfloor n/2 \rfloor \tag{54}$$

$$x_j = 0 \qquad \text{if } \gamma_j < \lfloor n/2 \rfloor. \tag{55}$$

   Next, obtain $z(n)$, the optimal objective function value of this problem.
2. From $k = n-1$ to $k = 1$ set the following valid inequalities:

$$\left\lceil \frac{k}{k+1} z(k+1) \right\rceil \leq z(k) \leq z(k+1) - d_{(k+1)}(x(k+1)).$$

From the discussion above, it is clear that, after solving the problem with $W(k+1)$, we have the lower bound (53) on $z(k)$ that we can use as a valid inequality when solving the problem with weighting vector $W(k)$. On the other hand, (52) provides an upper bound on $z(k)$ by $z(k+1)$. We will show in our computational experiments section that the improvements obtained by the application of these strategies are remarkable (see Section 5). Indeed, our computational experiments show that, even if the aim would be solving a problem for a given k', it is worth solving before the same instance for $k = n, \ldots, k'+1$ since, using the bounds and the initial solution provided for each k on the problem for $k-1$, produces an overall time saving. We observe that solving directly the problem for $k = k'$, without exploiting the result in Proposition 2, is more time consuming than solving the problems for $k = n, \ldots, k'+1$ by the iterative scheme illustrated above (derived from Proposition 2). This can be easily checked by adding up all the computing times needed for solving all problems involved in the above scheme up to k', and comparing the resulting computing time to the cpu time required to solve a single problem for the given k'. Actually, it may be even worse since, in many cases, solving directly the problem for k' may lead to reaching the maximum allowed computing time without having found even a feasible solution. On the other hand, the approach based on Proposition 2 solves the problem rather efficiently.

5. Computational results

This section reports on the results of an exhaustive computational test carried out on two sets of instances and our three formulations with and without implementing the improvement strategies illustrated in Section 4. We have tested data sets generated according to the scheme proposed in LeGrand et al. (2007). That paper distinguishes between *uniform* and *biased* data. Uniform data are obtained by generating all profiles as follows. For each element in the profile (candidate), set 1 (the candidate is approved) or 0 (not approved) according to a random variable that assumes values 0 and 1 with equal probability. Bias data are obtained by generating three sets of profiles separately. First, two possible approval probabilities, denoted by π_1 and π_2, are generated by selecting them from a random variable that is distributed uniformly in [0, 1]. Then, for the 40% of the profiles each element is generated by a random variable that assumes value 1 with probability π_1 and value 0 with probability $1 - \pi_1$. Another 40% is generated in the same way, but using π_2 in place of π_1. The remaining 20% is generated as in the case of uniform data (see LeGrand et al., 2007). Overall, we have solved 22,750 instances with the different combinations for n, m, k, and uniform and biased data.

Table 3
Valid inequalities generated in a Branch & Bound tree for solving formulation (25)–(27) with $n = 50$ and for uniform and biased data.

Data	Avg # Cuts	Max # Cuts	Avg # Cuts node	Max # Cuts node
Uniform	97,963.15	9,428,791	124.43	463
Biased	7390.41	315,094	63.80	444

In a preliminary analysis, we wanted to test the performance of our three formulations in instances of small size ($n = 50$ and different values for $m = 30, 35, 40, 45, 50, 55, 60$), in order to make the decision of which are the formulations to be tested on instances of larger size. Tables 1 and 2 report the average results of all the 1750 instances for $n = 50$ for the uniform and biased data (5 randomly generated instances for each m and $k = 1, \ldots, 50$). The results are organized as follows. Each row reports information about a different formulation that is indicated by its references. All instances were solved to optimality by all formulations. For each formulation we include information about average and maximum time expressed in seconds (Time (s)), average and maximum percent gap at the root node (GAP (%)), average and maximum number of nodes in the searching tree (Nodes), percentage of instances solved at the root node (%Solved root) and percentage of binary variables fixed with the preprocessing (%Fixed).

For formulation (25)–(27), we also report information on the average and maximum number of cuts (# Cuts), and the average and maximum number of cuts in each node (# Cuts node) (see Table 3). This information is relevant to understand the number of constraints, out of the exponentially many in the formulation, which are needed to have a valid representation of the problem at each node of the Branch & Bound tree.

We have also tested empirically (see Tables 1 and 2), that the gap at the root node coincides for the three formulations. This confirms that the three formulations are equivalent in terms of LP gap

Table 4
Summary for n=100, uniform data.

	Form.	Time (s)		GAP (%)		Nodes		%Solved root	% Fixed
		Avg	Max	Avg	Max	Avg	Max		
m=30	(41)–(47)	3.35	23.66	0.17	0.84	1565.53	28,751	46.0	9.32
	(33)–(38)	0.37	4.04	0.17	0.84	1488.07	36,793	46.4	9.32
m=35	(41)–(47)	4.39	55.32	0.17	2.50	2695.97	12,6951	43.8	8.87
	(33)–(38)	0.48	10.47	0.17	2.50	2334.13	96,381	44.8	8.87
m=40	(41)–(47)	5.40	93.11	0.14	1.14	4149.20	158,828	44.4	8.42
	(33)–(38)	0.70	12.53	0.14	1.14	3674.94	121,300	45.0	8.42
m=45	(41)–(47)	13.40	251.75	0.13	2.00	11,463.60	657,000	42.6	9.44
	(33)–(38)	1.56	65.41	0.13	2.00	9924.58	578,547	43.6	9.44
m=50	(41)–(47)	18.30	532.31	0.12	0.93	19,495.08	781,787	43.6	12.38
	(33)–(38)	2.48	78.21	0.12	0.93	17,922.55	605,694	45.2	12.38
m=55	(41)–(47)	57.27	1652.28	0.10	0.49	53,060.66	1,875,622	41.0	7.55
	(33)–(38)	5.12	131.90	0.10	0.49	45,278.16	1,256,747	40.2	7.55
m=60	(41)–(47)	41.62	2306.34	0.11	1.56	68,105.88	7,640,315	40.0	10.30
	(33)–(38)	5.78	382.24	0.11	1.56	47,840.94	4,036,987	39.6	10.30

Table 5
Summary for n=100, biased data.

	Form.	Time (s)		GAP (%)		Nodes		%Solved root	%Fixed
		Avg	Max	Avg	Max	Avg	Max		
m=30	(41)–(47)	0.58	1.96	0.30	3.13	48.83	2215	48.2	28.33
	(33)–(38)	0.08	0.42	0.30	3.13	44.13	2013	48.6	28.33
m=35	(41)–(47)	0.63	2.02	0.28	2.63	77.93	4147	43.6	30.07
	(33)–(38)	0.10	0.93	0.28	2.63	81.37	6788	45.6	30.07
m=40	(41)–(47)	0.76	2.47	0.25	2.38	193.00	9921	42.0	26.62
	(33)–(38)	0.13	0.91	0.25	2.38	177.40	8989	48.4	26.62
m=45	(41)–(47)	0.69	2.23	0.24	2.17	145.96	8026	42.4	27.24
	(33)–(38)	0.12	0.81	0.24	2.17	121.17	5985	46.0	27.24
m=50	(41)–(47)	0.69	2.93	0.23	2.00	263.15	15,166	44.6	29.21
	(33)–(38)	0.13	0.93	0.23	2.00	202.81	6125	50.2	29.21
m=55	(41)–(47)	0.81	6.33	0.21	1.23	497.78	25,672	46.6	28.69
	(33)–(38)	0.17	3.33	0.21	1.23	443.27	26,845	52.2	28.69
m=60	(41)–(47)	1.06	39.10	0.22	1.67	1187.28	221,519	35.4	28.51
	(33)–(38)	0.26	24.92	0.22	1.67	1136.49	234,681	39.0	28.51

and explains the rather small integrality gaps of these formulations.

The conclusion of the above tables is that formulation (25)–(27) is as stronger as the other two in terms of gap, but its performance is inferior in terms of time and number of problems solved at the root node. For this reason, we have decided to carry out the final test for larger instances only for formulations (33)–(38) and (41)–(47). Therefore, we evaluate and compare the performance of formulations (33)–(38) and (41)–(47) on medium size instances, i.e., those with $n = 100, 200$.

Tables 4 and 5 report our results for the two types of data sets, i.e., uniform and biased, for $n = 100$. All the information is organized as in previous Tables 1 and 2. Again, all problems are solved at optimum.

The conclusions from Tables 4 and 5 are the following. Formulation (33)–(38) is one order of magnitude faster than (41)–(47). For instance, the average time for the largest instance sizes ($n = 100, m = 60$), solved with (33)–(38), is about 5.78 s and the maximum cpu time is 382.24 s. The same instances solved with (41)–(47) take an average and maximum time of 41.62 and 2306.34 s, respectively. This fact can be explained by the smaller number of variables and constraints required by formulation (33)–(38) with respect to (41)–(47). The remaining factors (GAP, Nodes, %Solved and %Fixed) are quite similar in both cases. In fact, as we have seen theoretically, both formulations are equivalent in terms of LP gap and they always fix the same number of binary variables. It is also very interesting to test the practical performance of a naïve approximation algorithm based on using the solutions of the linear relaxation, as proposed for instance in Amanatidis et al. (2015). One can easily bound from above the empirical performance ratio, $\frac{LPval}{optval}$, of any of our formulations based on the relative gap ($Rgap := \frac{optval - LPval}{optval} * 100\%$). Indeed, $\frac{optval}{LPval} \leq \frac{100}{100-Rgap}$. Actually, since the largest relative gap is below 3.13% (see Table 4), this results, in the worst case ($n = 100,\ m = 30$), with an empirical performance ratio bounded above by 1.03.

Next, we further test our best formulation, namely (33)–(38), on instances with increasing size in order to explore the size limit that can be solved within the time limit of 7200 s. Tables 6 and 7 report our results. As can be observed in these tables, there is a difference in the performance with respect to uniform and biased data. For uniform data we get the optimum within the fixed time limit already for $n = 150$. For biased data we are able to solve all instances to optimality within the time limit with instance size up to $n = 200$. The reason for this clear difference relies on the fact that the preprocessing (50) and (51) is much more efficient for biased data where a greater percentage of variables are fixed either to zero or to one. Indeed, this percentage is on average of 24.79% for biased data with $n = 200$ as compared to only 7.34% for the uniform data for $n = 150$.

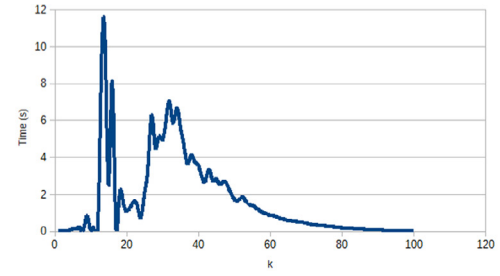
In order to analyze the performance of formulation (33)–(38), in Fig. 1 we visualize two measures of efficiency computed for the different $k = 1, \ldots, 100$. For each k, we show the cpu time required on average for solving the 35 instances with $n = 100$ (5 randomly generated instances for each value of $m \in \{30, 35, 40, 45, 50, 55, 60\}$). We also provide the number of instances solved at the root node (without branching). We have observed that the behavior is similar for the different values of n. For this reason, we show only the plots related to the case of $n = 100$. Fig. 1a and b refer to uniform data, while Fig. 1c and d refer to biased data.

Table 6
Summary for n=200, biased data (solved to optimality within 7200 s).

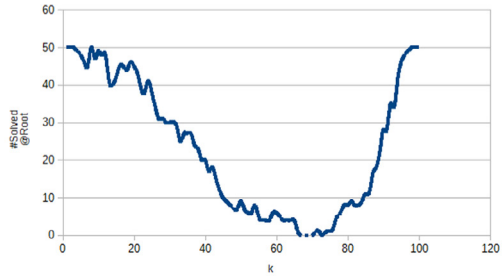
	Form.	Time (s)		GAP (%)		Nodes		%Solved root	%Fixed
		Avg	Max	Avg	Max	Avg	Max		
m=30	(33)–(38)	0.44	9.12	0.12	1.85	819.95	35,683	60.6	29.83
m=35	(33)–(38)	2.80	39.15	0.12	1.59	7962.11	162,887	38.5	14.09
m=40	(33)–(38)	6.27	225.75	0.10	1.59	22,843.11	1,165,252	56.1	24.90
m=45	(33)–(38)	21.63	985.73	0.09	1.33	76,974.33	4,847,789	48.6	27.74
m=50	(33)–(38)	42.09	1129.91	0.08	0.93	143,305.51	4,607,534	41.7	24.76
m=55	(33)–(38)	50.93	5070.65	0.09	1.61	186,926.77	28,259,032	40.7	20.79
m=60	(33)–(38)	49.97	5409.05	0.08	0.81	207,304.41	29,489,376	56.8	31.45

Table 7
Summary for n=150, uniform data (solved to optimality within 7200 s).

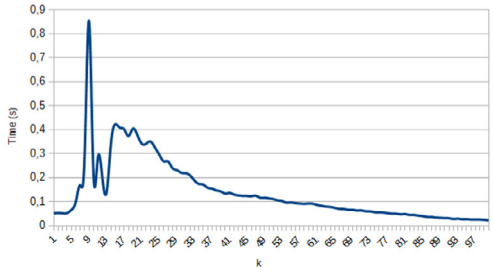
	Form.	Time (s)		GAP (%)		Nodes		%Solved root	%Fixed
		Avg	Max	Avg	Max	Avg	Max		
m=30	(33)–(38)	1.20	11.01	0.15	0.93	4302.39	60,713	28.27	7.44
m=35	(33)–(38)	2.74	45.11	0.15	2.38	11,338.78	237,964	30.80	6.97
m=40	(33)–(38)	7.87	138.90	0.13	0.62	36,490.51	768,328	28.80	7.43
m=45	(33)–(38)	12.64	302.69	0.12	1.92	56,540.63	1,886,774	27.33	7.61
m=50	(33)–(38)	44.67	777.99	0.11	0.89	211,780.07	4,787,806	28.67	7.43
m=55	(33)–(38)	81.99	4068.90	0.10	1.08	419,709.27	27,576,236	23.87	7.48
m=60	(33)–(38)	209.87	6664.97	0.10	0.76	1,023,055.10	36,286,662	24.67	6.99



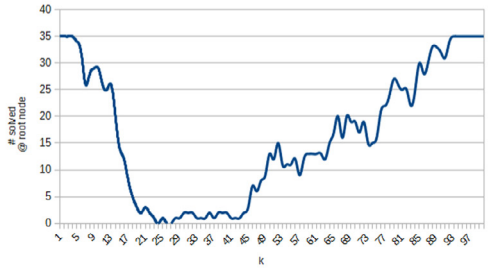
(a) Cpu time (in seconds) - uniform data.



(b) Instances solved at the root node - uniform data.



(c) Cpu time (in seconds) - biased data.



(d) Instances solved at the root node - biased data.

Fig. 1. Efficiency measures vs. k for formulation (33)–(38) with $n = 100$.

Analyzing the figures, we conclude that the general behavior is rather similar for uniform and biased data. Regarding the computing time for solving the problems, we observe that it increases when k decreases from $k = n$ until a certain threshold (which depends of the type of data, namely $k \in (15, 30)$ for uniform data and $k \in (9, 20)$ for biased data) and then the time decreases with k until $k = 1$. This trend can be explained from the combinatorics of the objective function which relates to $\binom{n}{k}$. With respect to the number of instances solved at the root node, the performance is also similar. This number decreases with k from $k = n$ until a certain value (which again depends on the type of data, $k \approx 70$ and $k \approx 28$, for uniform and biased data, respectively) and then it increases with k up to $k = 1$.

Focusing on Fig. 1a and c, one can also find additional evidence of the usefulness of the procedure following from Proposition 2 and discussed at the end of Section 4.

6. Minimizing the *h* smallest distances

In this section, we briefly discuss the problem of minimizing the sum of the h smallest Hamming distances already introduced in Section 2. In Amanatidis et al. (2015) this problem has been already considered in the approval voting application context, and the authors prove that, when the OWA vector is non-decreasing, that is, the weighting vector is of the form $M(n-h) = (0, \ldots, 0, 1, \ldots, 1)$, with h the number of ones, $1 \leq h < n$, the win-

ning committee can be found in polynomial time for a fixed value of h. They suggest an enumerative approach based on the solution of $\binom{n}{n-h} = \binom{n}{h}$ minisum problems that is obviously not efficient even for a fixed h, and not polynomial if h is part of the input. Using arguments similar to those in Section 2 for the problem of minimizing the sum of the k largest Hamming distances, also the problem of minimizing $M(n-h)$ for a fixed h (see Amanatidis et al. (2015)) can be proved to be polynomial applying Linear Programming.

For a fixed h, the general problem can be stated as

$$\min_{x\in\{0,1\}^m} \Big(\min \big\{ d_S(x) \mid S \subset \{1,\dots,n\}, |S| = h \big\} \Big). \tag{56}$$

We can switch the two *min* operators, thus obtaining

$$\min_{S\subset\{1,\dots,n\},|S|=h} \Big(\min_{x\in\{0,1\}^m} \big\{ d_S(x) \big\} \Big). \tag{57}$$

For a given subset $S \subset \{1,\dots,n\}$, the inner minimum in problem (57) corresponds to the minisum problem

$$\min \sum_{j=1}^{m} \gamma_j(S)(1-x_j) + \sum_{j=1}^{m} (h - \gamma_j(S)) x_j$$

$$\text{s.t. } 0 \le x \le 1,$$

which is polynomially solvable. Then, considering all the $\binom{n}{n-h} = \binom{n}{h}$ subsets of cardinality $n-h$, for a fixed h, the problem can be solved by a sequence of LPs.

7. Concluding remarks

In this paper we studied two versions of the k-sum approval voting problem, one minimizing the sum of the k largest distances from a voter profile to a committee, the other minimizing the sum of the h smallest distances. The latter approach for approval voting elections was not deeply analyzed in the literature, but it is worth remarking that its application in this context is meaningful. In fact, the problem can be stated as follows: elect a committee minimizing the sum of the h smallest Hamming distances from the voters' profiles. As already observed in Amanatidis et al. (2015), the application is significant for small values of $n-h$, say $n-h=1$ or $n-h=2$. Actually, in these cases, the assumption is that the first one or two maximum distances do not play a significant role in the selection of the committee, and this is true especially when the population of voters is extremely large. The idea is that there will be always some voters whose preferences are completely disjoint from those of the majority of the others. This is, in fact, a way of considering such voters' profiles as *outliers*. But, in our opinion, there are additional cases in which the application is meaningful, namely, for every choice of h such that $n-h \le \frac{n}{2}-1$. Under this condition, the approval voting problem consists of taking into account only the preferences of the absolute majority of the voters ($h \ge \frac{n}{2}+1$), with the aim of selecting the committee corresponding to the boolean vector x^* for which the sum of the Hamming distances w.r.t. the h considered profiles is minimized.

The two problems following the OWA approaches for approval voting studied in this paper can be seen as two different ways of facing the same problem, but giving more importance to one of the two principles that are at the basis of any democratic election. The problem with weighting vector $W(k) = (1,\dots,1,0,\dots,0)$ implements the idea that *representation* must be maximized. If, on the other hand, one wants to give more importance to *governability*, the minisum approach (with weighting vector $M(n-h) = (0,\dots,0,1,\dots,1)$) can be pursued with a suitable choice of h, since it is able to guarantee a strong and cohesive consensus. This strength can be enforced by increasing the value of h. We leave the choice of which is the best voting rule for a country to its lawmakers, who will be able to choose the best rule, according to the specific political and social situation in which the election takes place.

Going back to our theoretical considerations, to the best of our knowledge, the computational complexity of the minsum problem with weighting vector $M(n-h) = (0,\dots,0,1,\dots,1)$ when h is part of the input is still an open problem. In our opinion this is an interesting issue that will be the focus of our future work.

Acknowledgments

This research has been partially supported by the Spanish Ministry of Economy and Competitiveness/FEDER grants numbers MTM2013-46962-C02-01 and MTM2016-74983-C02-01. This paper has been written during a sabbatical period of the second author in Universitá di Roma La Sapienza whose support is also acknowledged.

The authors would like to thanks the referees for their interesting and constructive suggestions which help to improve the quality of the paper.

References

Amanatidis, G., Barrot, N., Lang, J., Markakis, E., Ries, B., 2015. Multiple referenda and multiwinner elections using hamming distances: complexity and manipulability. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, AAMAS, pp. 715–723.

Aziz, H., Gaspers, S., Gudmundsson, J., Mackenzie, S., Mattei, N., Walsh, T., 2015. Computational aspects of multi-winner approval voting. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, AAMAS, pp. 107–115.

Barrot, N., Lang, J., Yokoo, M., 2017. Manipulation of hamming-based approval voting for multiple referenda and committee elections. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS, pp. 597–605.

Blanco, V., Ali, S.E.H.B., Puerto, J., 2014. Revisiting several problems and algorithms in continuous location with ℓ_τ norms. Comput. Optim. Appl. 58, 563–595.

Brams, S.J., Fishburn, P., 1978. Approval voting. Am. Polit. Sci. Rev. 72, 831–847.

Brams, S.J., Fishburn, P., 2005. Going from theory to practice: the mixed success of approval voting. Soc. Choice Welfare 25, 457–474.

Brams, S.J., Fishburn, P., 2007. Approval Voting. Springer.

Brams, S.J., Kilgour, D.M., Sanver, M.R., 2007. A minimax procedure for electing committees. Public Choice 132, 401–420.

Byrka, J., Sornat, K., 2014. PTAS for minimax approval voting. In: 10th Conference on Web Interactions and Network Economics. Beijin.

Caragiannis, I., Kalaitzis, D., Markakis, E., 2010. Approximation algorithms and mechanism design for minimax approval voting. AAAI.

Fishburn, P., Pekec, A., 2004. Approval Voting for Committees: Threshold Approaches. Technical Report, 2004.

Frances, M., Litman, A., 1997. On covering problems of codes. Theory Comput. Syst. 30, 113–119.

Givant, S., Halmos, P., 2009. Introduction to Boolean Algebra. Springer.

Kilgour, D.M., 2010. (Chapter 6 of the handbook), approval balloting for multi-winner elections. In: Laslier, Sanver (Eds.), Handbook on approval voting. Springer. Chapter 6

Laslier, J. F., Sanver, M. R. (Eds.), 2010. Handbook on Approval Voting, Springer.

LeGrand, R., Markakis, E., Mehta, A., 2007. Some results on approximating the minimax solution in approval voting. AAMAS 07. Honolulu.

Li, M., Ma, B., Wang, L., 2002. On the closest string and substring problems. J. ACM 49, 157–171.

Menezes, M.B.C., Silveira, G.J.C.d., Drezner, Z., 2016. Democratic elections and centralized decisions: condorcet and approval voting compared with median and coverage locations. Eur. J. Oper. Res. 253, 195–203.

Nemhauser, G.L., Wolsey, L.A., 1988. Integer Programming and Combinatorial Optimization. Wiley, Chichester.

Nickel, S., Puerto, J., 2006. Location Theory: A Unified Approach. Springer.

Ogryczak, W., Tamir, A., 2003. Minimizing the sum of the k largest functions in linear time. Inf. Process. Lett. 85, 117–122.

Puerto, J., Rodríguez-Chía, A.M., Tamir, A., 2017. Revisiting k-sum optimization problems. Math. Prog. 165, 579–604.

Rapoport, A., Felsenthal, D.S., 1990. Efficacy in small electorates under plurality and approval voting. Public Choice 64, 57–71.

Taylor, A.D., Pacelli, A., 2008. Mathematics and politics. Springer Science+Businness Media, LLC, New York, second ed..

Yager, R.R., 1988. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Trans. Syst. Man Cybern. 18, 183–190.