Auton Agent Multi-Agent Syst
DOI 10.1007/s10458-010-9133-6

Practical voting rules with partial information

Meir Kalech · Sarit Kraus · Gal A. Kaminka · Claudia V. Goldman

Abstract Voting is an essential mechanism that allows multiple agents to reach a joint decision. The joint decision, representing a function over the preferences of all agents, is the winner among all possible (candidate) decisions. To compute the winning candidate, previous work has typically assumed that voters send their complete set of preferences for computation, and in fact this has been shown to be required in the worst case. However, in practice, it may be infeasible for all agents to send a complete set of preferences due to communication limitations and willingness to keep as much information private as possible. The goal of this paper is to empirically evaluate algorithms to reduce communication on various sets of experiments. Accordingly, we propose an iterative algorithm that allows the agents to send only part of their preferences, incrementally. Experiments with simulated and real-world data show that this algorithm results in an average of 35% savings in communications, while guaranteeing that the actual winning candidate is revealed. A second algorithm applies a greedy heuristic to save up to 90% of communications. While this heuristic algorithm cannot guarantee that a true winning candidate is found, we show that in practice, close approximations are obtained.

Keywords Multi-agent systems · Computational social choice · Voting

M. Kalech (✉)
Information Systems Engineering, Ben-Gurion University, Beer-Sheva, Israel
e-mail: kalech@bgu.ac.il

S. Kraus · G. A. Kaminka
Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel

C. V. Goldman
Samsung Telecom Research, Herzliya, Israel

Published online: 07 May 2010

1 Introduction

Voting is an essential mechanism that allows multiple agents to reach a joint decision, which reflects the preferences of all agents [3,13]. A voting protocol is a function that determines a winning decision among all candidate decisions. Its inputs are the preferences of the agents. Its output is the decision regarding the winning candidate, which commits all participating agents. As such, voting is fundamental to multi-agent systems.

Previous work on voting mechanisms have typically assumed that the voters provide their complete set of preferences to the voting protocol. Indeed, Conitzer and Sandholm [3] show that for many of the voting protocols, in the worst case one cannot do better than simply letting each voter communicate all of its preferences, so as to form a total order over the candidates' decisions.

However, in practice, several reasons conflict with the requirement for all voters to send their complete set of preferences to the voting center agent (hereinafter "center"), calculating the votes:

- First, bandwidth and communications costs may make it impractical to send the entire set of preferences at once.
- Second, sending the entire preference set requires having the voter actually ranking its preferences over the entire set of candidates. In practice, voters may wish to rank candidates only as necessary (lazy commitment), since that can be less expensive or time-consuming than the complete ranking.
- Third, some voters may intermittently offline; if they miss the single opportunity to vote at the beginning of the process, they will be completely cutoff.
- Finally, sending the entire preference set may violate the sender's privacy with respect to the center agent, collecting the votes (although there are existing efforts that independently tackle this particular concern [1,18]).

As a concrete example, consider a meeting scheduling application, in which human users vote on meeting times (e.g., such as [14]). It can be difficult for the user to specify preferences for all possible slots as there are hundreds in a month: Picking the best three slots is much easier for the human user than ranking all possible slots. It is also costly to communicate all preferences to the agent conducting the vote. Not all users are constantly online, and some may be late entering the voting process, and would thus like to seamlessly add their preferences. Finally, sending the entire set of preferences essentially blocks parallelization of meeting scheduling: Users must wait for one vote to finalize before specifying their entire set of preferences for another meeting.

In practice, of course, humans often schedule meetings without requiring the full set of preferences to be sent by all attendees. Instead, they vote incrementally, providing only a subset of their preferences at a time, and only communicating additional votes if needed. In other words, the theoretical worst case scenario as discussed in [3] is not necessarily common in practice.

This raises the challenge of *partial-order voting*, where agents only supply a subset of their preferences at any given time. Unfortunately, previous investigations have mainly focused on theoretical understanding of partial-order voting, without providing a complete protocol [3,12]. Konczak and Lang [8] present an algorithm which attempts to compute *possible winning candidates*, given a subset of preferences. It also attempts to find *inevitably winning candidates*, if those can be found using partial information. Pini et al. [11] consider the complexity of this process, and generalize it. Other related investigations are discussed in the next section. None of the investigations provides an evaluation using real-world data.

The primary goal of this paper is to introduce practical partial-order voting protocols to reduce communication by iteratively aggregating agents' preferences. We prove the properties of these algorithms analytically, and empirically evaluate their ability to reduce communications in various sets of experiments in simulation and real-world domains.

In particular, in Sect. 3 we present an *i*terative algorithm that allows the agents to incrementally send their preferences, converging to a winning candidate. The algorithm proceeds iteratively, by asking the agents to send the value of only one candidate each iteration. The value of a candidate is its score based on the voting protocol. For instance, in Range voting the score is taken from a pre-defined range, in Borda the score is the candidate's place in the candidates vector $(m-1, m-2, \ldots, 0)$; m being the number of candidates. The agents send the candidates in a decreasing order of their preference values. After each iteration, utilizing the technique outlined in [8], the center receiving the preferences of the agents checks whether enough information has been sent to declare a winner. We present and prove the correctness of this algorithm for various voting rules: for rank-based rules (Borda, Copeland, Maximin, Bucklin), and for non-ranking rules (Range, Approval, and Cumulative).

To further reduce the preferences communicated by the agents, we also present a *g*reedy-heuristic algorithm in Sect. 4. This algorithm significantly reduces the number of candidates' preference values that are sent. It stops after only a few iterations and chooses the most preferred candidates. Consequently it cannot guarantee that it will find the winner that would have been determined in a voting with full information. However, we show that in practice it works well.

Experiments with simulated and real-world data on Borda and Range rules are presented in Sect. 5. The experiments show that the iterative algorithm saves much communication while guaranteeing that a winning candidate will be found. In particular it saves 18–52% of the communication depending on the distribution of the agents, and 25–45% depending on the size of agents group. The heuristic algorithm saves up to 90% of communications, though it does not guarantee that a true winning candidate will be found. Nonetheless, the results show that the quality of the candidate it finds, compared to the real winning candidate, is very high. In addition, the experiments show that mainly two factors influence the communication of the iterative algorithm and the quality of the winner in the heuristic algorithm: (1) the homogeneity of the agents (the similarity of the preference values of the agents over the candidates), and (2) the diversity among the candidates' preference values.

2 Related work

There has been work that addresses voting in systems where agents send partial information, but most of them do not propose algorithms to reduce communication, and focus instead on the use of the partial information in static form. Furthermore, while most previous work proposes theoretical analysis of solving a voting problem, we evaluate our algorithms empirically.

Conitzer and Sandholm [3], analyze the communication complexity of various voting protocols and determine upper and lower bounds for communication. In general, they show that for most of the voting protocols in the worst case the agents should send the entire set of preferences for the candidates.

Konczak et al. [7,8] address the case of partial information, where the agents do not set the preferences for all the candidates. In this case they show how to compute the sets of **possible** winners and **necessary** winners. These sets determine which of the candidates are inevitably pruned from the candidate set and which are necessary winners. We adopt their approach to propose a systematic preference aggregation protocol in which the agents do not need to send their entire set of preferences.

Walsh [15] surveys the computational complexity of possible and necessary winners in various of voting rules. In that paper Walsh studies this problem along two dimensions: weighted or unweighted votes, and a bounded or unbounded number of candidates. Xia and Conitzer [17] continue to investigate this computational analysis where the votes are unweighted and the number of alternatives is not bounded in more rules other than STV rule.

Walsh [16] considers the question of how to decide when to stop eliciting preferences as the winner is guaranteed. Obviously this question relates to the possible and necessary winners computation. He shows that the computational complexity of deciding when to stop elicitation is affected by the way the agents elicit their preferences.

Pini et al. [11] generalize the problem to address incomparability between candidates beyond regular relations and unknown relations. They prove that the computational complexity of computing possible winners and necessary winners is polynomial for voting rules which are monotonic and independent of irrelevant alternatives (IIA). They present a polynomial algorithm to determine the winners by eliciting preferences. However, they do not suggest how to select the next candidate to ask about in order to reduce **communication**. Also they do not empirically evaluate their algorithm. We, on the other hand focus on the evaluation of communication. Thus we present iterative voting procedures both for Range voting that satisfies IIA, as well as to Borda that does not satisfy IIA. We present experimental results that evaluate our algorithms.

Gelain et al. [5] consider a similar problem of soft constraint problems where some of the preferences may be unspecified. They propose a branch and bound algorithm in which the user reveals as few preferences as possible to find an optimal solution. This work assumes that the agents provide a complete set of candidates but a partial set of preferences. In contrast, we propose incremental algorithms in which the candidate's set is not necessarily known in advance to the center. In addition, we propose a greedy heuristic which significantly reduces the number of revealed preferences.

Lang et al. [9] focus on majority voting and show how to apply sequential majority comparisons with an unknown tree. The idea of sequential comparison is similar to the iterative algorithms in this paper; however, Lang et al. focus on majority voting, while we present our approach using two other voting rules. Furthermore, the major contribution of our paper which goes beyond Lang et al.'s work and other previous work, is the empirical evaluation of our iterative voting procedures.

The idea of reducing communication by communicating the preferences in a decreasing order has been proposed in other contexts. For instance, Hudson and Sandholm [6] propose in combinatorial auction that the elicitor will ask the bidders for the next-most valuable bundle, based on the assumption that each bidder ranks its bundles in order of decreasing valuation. Faltings and Macho-Gonzalez [4] also present a model of open constraint optimization problems where the agents submit their preferences in a decreasing order to find an assignment to a set of variables that is optimal with respect to their preferences. As far as we know, no previous work has proposed this idea for voting systems.

3 Iterative voting procedure

In a regular voting procedure, voters submit to the center a complete set of their preference values for a set of candidates. The center calculates the winners based on the voting protocol. Formally, given a set of voter agents $A = \{a_1, \ldots, a_n\}$, and a set of candidates

$C = \{c_1, \ldots, c_m\}$, a voting protocol defines a winner determination function from the agents' votes to the set of candidates [2].

We present *iterative voting* algorithms to minimize the number of preference values the agents are required to submit in order to determine the winners. The key idea is that after the initial call for votes, the voters (agents) incrementally send their preference values *in rounds*, one preference each round in a decreasing order of their preferences. After each round, they wait for the center to decide whether sufficient information has been received to determine a winner, or additional candidates' preference values need to be sent. In each round the center calculates the subset of candidates that no longer have any chance of winning and prunes them from the candidate set. The iterative algorithm is terminated once the set of candidates includes only the winners, and from that point the agents no longer send preference values.

To implement voting in rounds, in each round we need to calculate the set of candidates that no longer have a chance of winning, and the set of candidates that must be winners based on the information sent so far. Based on these sets we can decide whether we can stop and declare the winners.

To do this, we adopt the definitions of possible and necessary winners provided by Konczak and Lang [8], for a given set of partially-ordered preferences. They define the set of **possible winners** as containing the candidates that win at least in **one** complete extension of the partial preference relations among the candidates. That is, a candidate C is a possible winner if there is at least one total order extension of the partial order of the preferences, in which C wins. Analogously, the **necessary winners** set contains the candidates that win in **all** complete extensions of the partial preference relations among the candidates.

It is unnecessary to generate all the complete extensions in order to check possible and necessary winners. It is sufficient to generate only the extensions that provide the most optimistic and pessimistic scenarios. For instance, in score-based voting rules (like Borda) a *possible winner* is a candidate whose maximum score is greater than the minimum score of all the other candidates, and a *necessary winner* is a candidate whose minimum score is greater than the maximum score of all the others.

The computation process of the possible and necessary winners, as well as the winner determination function depend on the voting protocol. In Sect. 3.1 below, we discuss iterative Range voting as representative of non-ranking voting rules. In Range voting (unlike, for instance, Cumulative) a voter can assign its scores to the candidates independently of each other. The independence of the assignments is very important in iterative voting where the voter assigns the candidates iteratively. In Sect. 3.2, we focus on iterative Borda voting as a representative of rank-based rules, since it is considered as a very common rule in the literature (see, e.g., [3,7–9,15–17]).

We have focused on Borda and Range rules since they differ in two attributes which influence the iterative algorithm:

1. Range voting satisfies the independence of irrelevant alternatives criterion, while Borda does not. This criterion determines that if c_i is preferred over c_j, then by changing the preference of a third candidate c_k, c_j must not be preferred over c_i. This criterion must be addressed in iterative voting since the voters do not explore their whole set of preferences, thus it is important whether a new preference of a candidate may affect the order of the other candidates.
2. Implementing the iterative algorithm with Borda rule requires the number of candidates to be known to the center while when implementing it with Range this information is not necessary. This requirement also differentiates between the other voting rules. For instance, the Copeland and Maximin voting rules require the center to know the number

of candidates while Bucklin and Cumulative do not. This point is important since there are voting systems in which the center does not necessarily know the set of candidates in advance.

Finally, in Sect. 3.3, we discuss iterative voting for other voting rules. We discuss other rank-based rules (Copeland, Maximin, Bucklin), and non-ranking rules (Approval, Cumulative). We show how iterative voting can be applied for each of these voting rules.

3.1 Iterative range voting

In Range voting, voters are asked to assign a point value within a specified range for each candidate. The candidate with the highest total number of points is the winner. Thus if an agent significantly prefers a candidate to the others, it will assign that candidate the highest value. On the other hand if a certain candidate is an impossible candidate for an agent, then it will assign that candidate the lowest value.[1] By providing the preference values in a decreasing order the user does not have to assign his preference values to all the candidates but only to those that he actually submits.

Let us formalize this process. In Range voting, the agents assign values to the candidates from a predefined domain $D = \{d_1, \ldots, d_k\}$, where d_1 stands for the lowest value and d_k denotes the highest. Agent a_i's preferences are represented by a scoring function $v^i : C \rightarrow D$ that assigns a value d_l to every candidate $c_j \in C$. $O_t^i = \{\langle c_p, d_q\rangle_1^i, \ldots, \langle c_r, d_s\rangle_t^i\}$ is a set of pairs, where $\langle c_j, d_k\rangle_f^i$ stands for $v^i(c_j) = d_k$ in iteration f. This set is maintained in a decreasing order of the candidates' preference values of agent a_i sent through iteration t. $\mathcal{O}_t^A = \{O_t^1, \ldots, O_t^n\}$ is a set of O_t^i sets.

As mentioned above, in score-based voting rules, in order to check the possible and necessary winners we must define the possible maximum and possible minimum scores of a candidate. In Range voting, the pessimistic value (possible minimum) of a candidate is the lowest bound of the range. The optimistic value (possible maximum) is the upper bound. However, in our iterative algorithm, we exploit the fact that the candidates are sent in a decreasing order, so the optimistic value of each new candidate must be equal or lower than the value of the candidates that have been priory submitted. The function $pmax_t^A(c_j, \mathcal{O}_t^A)$ (described in Algorithm 1) computes the possible maximum of candidate c_j, based on the preference values of the agents through iteration t:

Definition 1 (*Range voting Possible Maximum*) $pmax_t^A(c_j, \mathcal{O}_t^A) = \sum_i pmax_t^i(c_j, O_t^i)$, where

$$pmax_t^i(c_j, O_t^i) = \begin{cases} d & \text{if } \exists d : \langle c_j, d\rangle \in O_t^i \\ d_s, \text{ where } < c_r, d_s >_t^i \in O_t^i & \text{otherwise} \end{cases} \tag{1}$$

The meaning of $pmax$ function is, that for those agents that have already submitted c_j the value is the one they have assigned (d), while for those agents that have not sent it yet, the value is the last value they have sent (in iteration t) since the candidates are submitted in a decreasing order. In the same way we define a function of the possible minimum of candidate c_j through iteration t: $pmin_t^A(c_j, \mathcal{O}_t^A)$.

Definition 2 (*Range voting Possible Minimum*) $pmin_t^A(c_j, \mathcal{O}_t^A) = \sum_i pmin_t^i(c_j, O_t^i)$, where

$$pmin_t^i(c_j, O_t^i) = \begin{cases} d & \text{if } \exists d : \langle c_j, d\rangle \in O_t^i \\ d_1 & \text{otherwise} \end{cases} \tag{2}$$

[1] Obviously, though Range voting is very susceptible to manipulation, we assume no manipulation in this paper.

Algorithm 1 describes the calculation of the possible maximum of candidate c', in the Range voting protocol. The algorithm receives the candidate c' and $\mathcal{O}_t^A$ (the preference values of the agents through iteration t). The main loop reviews the agents, and for each one of them it searches for candidate c' (lines 4–9). If it is found (line 5) then it adds the value d of that candidate to the max and stops the search. If the candidate is not found, then it was not submitted by that agent and its maximum value is equal at most to the value of the last candidate of that agent (line 11). The calculation of the possible minimum is the same except in the case where c' was not submitted by the agent. In this case it adds the minimal possible value d_1 in domain D (line 11).

Algorithm 1 CALCULATE_MAX_RANGE

```
(input: candidate $c'$,
        ordered preference values of set $A$: $\mathcal{O}_t^A$
output: the max value of candidate $c'$)

1:  $max \leftarrow 0$
2:  for all $O_t^i \in \mathcal{O}_t^A$ do
3:    $flag \leftarrow NOT\_FOUND$
4:    for all $\langle c, d\rangle_f^i \in O_t^i$ and $flag == NOT\_FOUND$ do
5:      if $c' == c$ then
6:        $max \leftarrow max + d$
7:        $flag \leftarrow FOUND$
8:      end if
9:    end for
10:   if $flag == NOT\_FOUND$ then
11:     $max \leftarrow max + d_s,\ where\ < c_r, d_s >_t^i \in O_t^i$
12:   end if
13: end for
14: return $max$
```

Now that we have defined the possible maximum and minimum values of each candidate, we can define the possible winner and the necessary winner. For these definitions we define the set $C_t \subseteq C$ which contains the candidates that have been sent by any agent through iteration t.

Definition 3 (*Current Candidates*) $C_t = \{c | \langle c, d\rangle_j^i \in O_t^i,\ O_t^i \in \mathcal{O}_t^A\}$

A *possible winner* is a candidate whose maximum score is greater than the minimum score of all the other candidates:

Definition 4 (*Possible Winner*)

$$C_p = \{c_i | pmax_t^A(c_i, \mathcal{O}_t^A) \geq pmin_t^A(c_j, \mathcal{O}_t^A)\ \forall c_j \in C_t \setminus \{c_i\}\}$$

A *necessary winner* is a candidate whose minimum score is greater than the maximum score of all the others:

Definition 5 (*Necessary Winner*)

$$C_n = \{c_i | pmin_t^A(c_i, \mathcal{O}_t^A) > pmax_t^A(c_j, \mathcal{O}_t^A)\ \forall c_j \in C_t \setminus \{c_i\}\}$$

Algorithm 2 presents the iterative process which finds the necessary winners. This algorithm is invoked once the center of the voting initiates a request for a vote. In lines 1–3 the sets of the ordered preference values of the agents are initialized as well as the sets of the possible and necessary winners. Iteratively, the center receives a value from the agents representing their next preference value in a decreasing order (O^i_t in lines 6–9). Then it invokes the Algorithm CALCULATE_POSSIBLE_WINNERS in order to calculate the set of possible winners (line 11) and CALCULATE_NECESSARY_WINNERS to calculate the set of necessary winners (lines 12) based on the formulas given in Definitions 4 and 5, respectively.

The CALCULATE_POSSIBLE_WINNERS algorithm prunes the candidates whose maximal value is lower than at least one minimal value of another candidate. In a similar manner, the CALCULATE_NECESSARY_WINNERS algorithm prunes the candidates whose minimal value is lower than at least one maximal value of another candidate (these algorithms are not presented). The center agent stops the iterative process once the set of the possible winners contains only necessary winners (line 5) (in the first iteration it receives the candidates from the agents in any case).

Algorithm 2 ITERATIVE_NECESSARY_WINNERS
(**output**: set of necessary winning candidates C_n)

```
1: $t \leftarrow 1$
2: $O^1_t \leftarrow \emptyset, O^2_t \leftarrow \emptyset, \ldots, O^n_t \leftarrow \emptyset$
3: $C_p \leftarrow \emptyset$
4: $C_n \leftarrow \emptyset$
5: while $C_p - C_n \neq \emptyset \vee t = 1$ do
6:   for all $a_i \in A$ do
7:     receive $\langle c, d \rangle^i_t$
8:     $O^i_t \leftarrow O^i_t \bigcup \{\langle c, d \rangle^i_t\}$
9:   end for
10:   $\mathcal{O}^A_t \leftarrow \{O^1_t, O^2_t, \ldots, O^n_t\}$
11:   $C_p \leftarrow CALCULATE\_POSSIBLE\_WINNERS(\mathcal{O}^A_t)$
12:   $C_n \leftarrow CALCULATE\_NECESSARY\_WINNERS(\mathcal{O}^A_t)$
13:   $t \leftarrow t + 1$
14: end while
15: return $C_n$
```

Example 1 Consider a set $A = \{a_1, a_2, a_3\}$ and candidates set $C = \{c_1, c_2, c_3, c_4\}$. The domain of the candidates' value is $D = \{1, 2, 3, 4, 5\}$. The preference values of the agents are presented in Table 1. For instance, the third column in the first row is given by the function $v^1(c_3) = 4$. The algorithm proceeds in rounds such that in the first iteration the agents send only the highest valued candidate. a_1 sends $\langle c_1, 5\rangle$, a_2 sends $\langle c_2, 3\rangle$ and a_3 sends $\langle c_1, 4\rangle$.

Let us calculate the possible maximum and minimum values of the candidates, as computed by the center based on incomplete information it receives from the agents The center has received the value of candidate c_1 for agents a_1 (5) and a_3 (4), but has not received its value for agent a_2. However, it can infer that a_2's possible maximum value is 3, since this is the value it lastly submitted for c_2 (agents send their preference values in a decreasing order). Therefore, the possible maximum value of c_1 is $5+4+3 = 12$. Similarly, its possible

Table 1 The candidates' preference values of the agents in set $A = \{a_1, a_2, a_3\}$

$v^i(c_j)$	c_1	c_2	c_3	c_4
a_1	5	3	4	3
a_2	1	3	2	2
a_3	4	3	2	1

minimum value for candidate c_1 by agent a_2 is 1 since it is the minimum value in the domain. Therefore, the possible minimum value of c_1 is $5 + 4 + 1 = 10$. The possible minimum and maximum values of all the candidates after the first iteration ($t = 1$) are presented in the first row of Table 8.

In the second iteration a_1 sends $\langle c_3, 4\rangle$, a_2 sends $\langle c_4, 2\rangle$ (it randomly selects between c_3 and c_4) and a_3 sends $\langle c_2, 3\rangle$. The possible minimum and maximum values after the second iteration ($t = 1$) are presented in the second row of Table 8. Now, we can see that $pmax_2^A(c_3, \mathcal{O}_t^A) < pmin_2^A(c_1, \mathcal{O}_t^A)$ and $pmax_2^A(c_4, \mathcal{O}_t^A) < pmin_2^A(c_1, \mathcal{O}_t^A)$, thus, candidates c_3 and c_4 are no longer possible winners and they are pruned. Lastly, in the third row of Table 8 we can see that after the third iteration the center concludes that candidate c_1 is the necessary winner since $pmin_3^A(c_1, \mathcal{O}_t^A) > pmax_3^A(c_i, \mathcal{O}_t^A)$ where $i = \{2, 3, 4\}$. In this case, the agents do not send their last candidate since a necessary winner has been found.

Algorithm 2 finds the optimal winners; namely, the final set C_n is equal to the set of winners in a regular Range voting protocol where the agents send their whole set of candidates in one shot. In order to prove this statement we need to prove that any candidate that does not belong to the set of necessary winners (C_n) in the final iteration cannot be a winner. Formally:

Lemma 1 *Given the final iteration t^* in iterative Range voting,*

$$\forall c \in C \setminus C_n,\ \forall c' \in C_n : \ pmax_{t^*}^A(c, \mathcal{O}_{t^*}^A) < pmin_{t^*}^A(c', \mathcal{O}_{t^*}^A).$$

Proof:

1. Assume $c \in C_{t^*} \setminus C_n$ (c has been sent by one of the agents but is not a necessary winner) then c is pruned by definition of the necessary winner (definition 5).
2. Assume $c \in C \setminus C_{t^*}$ (c has not been sent by any agent) then we will prove that $pmax_{t^*}^A(c, \mathcal{O}_{t^*}^A) \leq pmax_{t^*}^A(c'', \mathcal{O}_{t^*}^A)$ where $c'' \in C_{t^*} \setminus C_n$.
   By definition, $pmax_{t^*}^A(c'', \mathcal{O}_{t^*}^A) = \sum_{i=1}^n pmax_{t^*}^i(c'', O_{t^*}^i)$. Since c has not been sent yet, and since the agents send their candidates in a decreasing order of their preference values $\forall i \in n : \ pmax_{t^*}^i(c, O_{t^*}^i) \leq pmax_{t^*}^i(c'', O_{t^*}^i)$. Therefore, $\sum_{i=1}^n pmax_{t^*}^i(c, O_{t^*}^i) \leq \sum_{i=1}^n pmax_{t^*}^i(c'', O_{t^*}^i)$, namely, $pmax_{t^*}^A(c, \mathcal{O}_{t^*}^A) \leq pmax_{t^*}^A(c'', \mathcal{O}_{t^*}^A)$. In the first part of the proof we proved that $\forall c \in C_t \setminus C_n$: $pmax_{t^*}^A(c, \mathcal{O}_{t^*}^A) < pmin_{t^*}^A(c', \mathcal{O}_{t^*}^A)$, therefore $pmax_{t^*}^A(c'', \mathcal{O}_{t^*}^A) < pmin_{t^*}^A(c', \mathcal{O}_{t^*}^A)$. □

Let $score(c)$ denote the final score of c in regular Range voting where the agents send their whole set of candidates' preference values in one iteration, then $score(c) = \sum_i v^i(c)$. The relation between $score(c)$ and the possible maximum and minimum of c during an iterative Range voting protocol is presented in the next lemma:

Lemma 2 *Given $\mathcal{O}_t^A$ - a decreasing order of the candidates' preference values of agents in set A sent through iteration t in iterative Range voting,*

$$\forall t : \ pmin_t^A(c, \mathcal{O}_t^A) \leq score(c) \leq pmax_t^A(c, \mathcal{O}_t^A)$$

Proof:

1. $score(c) \leq pmax_t^A(c, \mathcal{O}_t^A)$: By definition $score(c) = \sum_i v^i(c)$. If $v^i(c)$ has been submitted by agents a_i, then by Definition 1 $pmax_t^i(c, O_t^i) = v^i(c)$. Otherwise,

$pmax_t^i(c, O_t^i) = d \in \langle c, d\rangle_t^i$, where $\langle c, d\rangle_t^i$ is the last preference value that has been sent in O_t^i, thus $pmax_t^i(c, O_t^i) \geq v^i(c)$. Consequently, $\sum_i pmax_t^i(c, O_t^i) \geq \sum_i v^i(c)$. By Definition 1 $pmax_t^A(c, \mathcal{O}_t^A) = \sum_i pmax_t^i(c, O_t^i)$, and $score(c) \leq pmax_t^A(c, \mathcal{O}_t^A)$.

2. $pmin_t^A(c, \mathcal{O}_t^A) \leq score(c)$: If $v^i(c)$ has been submitted by agents a_i, then by Definition 2 $pmin_t^i(c, O_t^i) = v^i(c)$. Otherwise, $pmin_t^i(c, O_t^i) = d_1$, and thus $pmin_t^i(c, O_t^i) \leq v^i(c)$. Consequently, $\sum_i pmin_t^i(c, O_t^i) \leq \sum_i v^i(c)$. By Definition 2 $pmin_t^A(c, \mathcal{O}_t^A) = \sum_i pmin_t^i(c, O_t^i)$, and $pmin_t^A(c, \mathcal{O}_t^A) \leq score(c)$. □

Now, we can prove the next theorem:

Theorem 1 *Assume C_w is the set of winners in regular Range voting where the agents send their whole set of candidates' preference values in one iteration, then the set C_n in the final iteration t^* in iterative Range voting affirms: $C_n = C_w$.*

Proof: In regular Range voting $c' \in C_w\ iff\ \forall c \in C \setminus C_w : score(c) < score(c')$. Based on Lemma 1 $\forall c \in C \setminus C_n$ and $\forall c' \in C_n$ $pmax_{t*}^A(c, \mathcal{O}_{t*}^A) < pmin_{t*}^A(c', \mathcal{O}_{t*}^A)$. By Lemma 2 we know that $pmin_{t*}^A(c', \mathcal{O}_{t*}^A) \leq score(c')$ and that $pmax_{t*}^A(c, \mathcal{O}_{t*}^A) \geq score(c)$. We can conclude that $\forall c \in C \setminus C_n$ and $\forall c' \in C_n : score(c) < score(c')$, and consequently $\forall c' \in C_n : c' \in C_w$. □

3.2 Iterative Borda voting

In the **Borda** voting protocol, every voter ranks the m candidates. The score of the most preferred candidate is $m - 1$ points, the second $m - 2$, etc., and the score of the least preferred candidate is 0, where m is the size of the candidates' set. To determine the winner, the score of each candidate from the agents' votes is summed up and the winner is the candidate with the highest score. In contrast to Range voting, the user cannot assign its preferences numerically. Nonetheless, the voter still does not have to assign his preferences over all the candidates in advance, but can do it in rounds by submitting his preferences in a decreasing order as requested.

The Borda protocol does not satisfy the independence of irrelevant alternatives criterion, which means, that the information about a given candidate could change the preference order of two other candidates. Although, the center still does not have to know the set of candidates in advance, it does have to know the size of this set in order to determine the winners as we prove below.

Let us formalize this process. The tuple $O_t^i = \langle c_{p_1}^i, \ldots, c_{p_t}^i\rangle$ is a decreasing order of the candidates submitted by a_i through iteration t. $\mathcal{O}_t^A = \{O_t^1, \ldots, O_t^n\}$ is a set of O_t^i tuples ($i \in \{1, \ldots, n\}$) of the agents in set A.

As mentioned above, in order to check the possible and necessary winners we must define the possible maximum and possible minimum scores of a candidate. The score of candidate c_j in Borda is determined by its preference in relation to the other candidates. However, the center has information only about (1) the candidates that have been sent through iteration t ($\mathcal{O}_t^A$) and (2) the size of the candidate set ($|C|$). To calculate the possible minimum and maximum of candidate c_j for a certain agent a_i, we need to take into consideration two types of information: (1) the candidates that have been sent by a_i through iteration t (O_t^i), (2) The difference between the size of C and the size of O_t^i. Let $C_{t\setminus i}$ be the set of candidates that have not been submitted yet by agent a_i by time t, then $|C_{t\setminus i}|$ denote this difference: $|C_{t\setminus i}| = |C| - |O_t^i|$.

Based on the first type of information we can calculate the preference of c_i in relation to the candidates that have been sent by agent a_i through iteration t. This preference value is the same both for the possible maximum as well as for the possible minimum. In addition, we should add the possible score given by the second type of information. If $c^i_{p_j} \in O^i_t$ then $c^i_{p_j}$ must be preferred over $\forall c_k \in C_{t\setminus i}$, since a_i submits its candidates according to its preferences in a decreasing order. Therefore, the possible maximum and minimum of $c^i_{p_j}$ in this case would be $|C_{t\setminus i}|$. However, if $c^i_{p_j} \notin O^i_t$, then $c^i_{p_j}$'s possible maximum is $|C_{t\setminus i}| - 1$ (assuming $c^i_{p_j}$ is the most preferred in $C_{t\setminus i}$), and its possible minimum is 0 (assuming c_i is the least preferred candidate in $C_{t\setminus i}$).

Let $preference^i(c)$ be the score that candidate c obtains from agent a_i's vote in a full information Borda voting protocol, then the possible maximum and minimum are as follows:

Definition 6 (*Borda Possible Maximum*) $pmax^A_t(c_j, \mathcal{O}^A_t) = \sum_i pmax^i_t(c_j, O^i_t)$, where

$$pmax^i_t(c_j, O^i_t) = \begin{cases} preference^i(c_j) & \text{if } c_j \in O^i_t \\ |C| - |O^i_t| - 1 & \text{otherwise} \end{cases}$$

Definition 7 (*Borda Possible Minimum*) $pmin^A_t(c_j, \mathcal{O}^A_t) = \sum_i pmin^i_t(c_j, O^i_t)$, where

$$pmax^i_t(c_j, O^i_t) = \begin{cases} preference^i(c_j) & \text{if } c_j \in O^i_t \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Algorithm 3 presents the calculation of the possible maximum of c' in the Borda protocol. In lines 5–11 we compute the possible maximum of c' based on O^i_t. In particular, we search for c' in O^i_t. Once it is found, we determine its possible maximum as the size of the rest of the set O^i_t (line 7). Lines 12–16 address the influence of $|C_{t\setminus i}|$ on the maximum. If c' was found in O^i_t then we add $|C_{t\setminus i}|$ to the maximum, otherwise the possible maximum is the size of $C_{t\setminus i}$. To calculate the possible minimum, we can use the same algorithm, except for line 13. There, rather than adding $|C_{t\setminus i}| - 1$, we add 0 (i.e., not change the result).

Algorithm 3 CALCULATE_MAX_BORDA
(**input**: candidate c',
ordered preference values of set A: $\mathcal{O}^A_t$
output: the max value of candidate c')

```
1: $max \leftarrow 0$
2: for all $O^i_t \in \mathcal{O}^A_t$ do
3:   $|C_{t\setminus i}| \leftarrow |C| - |O^i_t|$
4:   $flag \leftarrow NOT\_FOUND$
5:   for $j = 1$ to $t$ do
6:     if $c' = c^i_{p_j} \in O^i_t$ then
7:       $max \leftarrow max + |O^i_t| - j$
8:       $flag \leftarrow FOUND$
9:       $break$
10:    end if
11:  end for
12:  if $flag = NOT\_FOUND$ then
13:    $max \leftarrow max + |C_{t\setminus i}| - 1$
14:  else
15:    $max \leftarrow max + |C_{t\setminus i}|$
16:  end if
17: end for
18: return $max$
```

Table 2 The possible maximum and minimum values of candidates $\{C_1, C_2, C_3, C_4\}$

t	c_1		c_2		c_3		c_4	
	max	*min*	*max*	*min*	*max*	*min*	*max*	*min*
1	12	10	12	5	N/A	N/A	N/A	N/A
2	11	10	10	7	9	7	9	7
3	11	10	9	7	N/A	N/A	N/A	N/A

Table 3 The preferences of the agents in set $A = \{a_1, a_2, a_3\}$ in Borda protocol

$v^i(c_j)$	c_1	c_2	c_3	c_4
a_1	1	2	3	0
a_2	0	3	1	2
a_3	3	2	1	0

Table 4 The possible maximum and minimum values of candidates $\{C_1, C_2, C_3, C_4\}$

t	c_1		c_2		c_3		c_4	
	max	*min*	*max*	*min*	*max*	*min*	*max*	*min*
1	7	3	7	3	7	3	N/A	N/A
2	5	3	7	7	5	3	4	2

The ITERATIVE_NECESSARY_WINNERS algorithm (Algorithm 2), described in Sect. 3 proceeds exactly in the same manner as the iterative Range voting. The only difference is that in Borda the center receives the agents' next candidate ($c^i_{p_t}$) rather than the preference value of the candidate (line 9: $\langle c, d\rangle^i_t$).

Example 2 Consider a set $A = \{a_1, a_2, a_3\}$ and a set of candidates $C = \{c_1, c_2, c_3, c_4\}$. The preferences of the agents are presented in Table 2. Notice that the preferences of the agents are arranged in a decreasing order of $m - 1$ to 0. The algorithm proceeds in rounds such that in the first iteration the agents send only the most preferred candidate. a_1 sends c_3, a_2 sends c_2 and a_3 sends c_1.

Let us calculate the possible maximum and minimum values of the candidates. Since a_1 has not submitted c_1, c_1's possible maximum according to a_1 is 2, given that in the optimistic extension c_1 will be the next preferred candidate of a_2. Since the center knows that $|C| = 4$ it can determine its possible maximum as $|C| - |O^1_1| - 1 = 2$. Its possible minimum is 0 given that in the pessimistic extension it will be sent last (Table 3). The same is true for agent a_2. a_3 has submitted c_1, we can conclude that it is preferred over the rest of the candidates in the set $|C| - |O^3| = 3$. Finally, the possible maximum of c_1 is $2 + 2 + 3 = 7$ and its possible minimum is $0 + 0 + 3 = 3$. For symmetric reasons these results are the same for c_2 and c_3 (presented in Table 4).

In the second iteration a_1 sends c_2, a_2 sends c_4 and a_3 sends c_2. The possible minimum and maximum values after the second iteration ($t = 1$) are presented in the second row of Table 4. a_1 still has not submitted c_1, thus its possible maximum is 1 and its possible minimum is 0. The same for a_2. The possible maximum of c_1 based on the preferences sent by a_3 remains 3. Finally, the possible maximum of c_1 is 5 and its possible minimum is 3. The same analysis is true for c_3. On the other hand, all the agents submitted their preferences over

c_2, and we can determine its value as 7 with certainty. Now that the possible maximum of c_1, c_3 and c_4 is less than the minimum of c_2 we can prune them and declare c_2 as a necessary winner.

By using the Borda protocol, Algorithm 2 finds the optimal winners. In order to prove this statement we need to prove, as in the Range voting case, that any candidate not belonging to the set of the necessary winners (C_n) in the final iteration cannot be a winner. Formally:

Lemma 3 *Given the final iteration t^* in iterative Borda voting,*

$$\forall c \in C \setminus C_n,\ \forall c' \in C_n :\ pmax^A_{t^*}(c, \mathcal{O}^A_{t^*}) < pmin^A_{t^*}(c', \mathcal{O}^A_{t^*}).$$

Proof: The same proof as for Lemma 1. □

Let $score(c)$ denote the final score of c in regular Borda voting, where the agents send their whole set of preferences in one iteration. Then $score(c) = \sum_i preference^i(c)$, where $preference^i(c)$ is the preference of candidate c by agent a_i. The relation between $score(c)$ and the possible maximum and minimum of c during an iterative Borda voting protocol is presented in the next lemma:

Lemma 4 *Given $\mathcal{O}^A_t$ - a decreasing order of the candidates' preference values of agents in the set A sent through iteration t in iterative Borda voting,*

$$\forall t :\ pmin^A_t(c, \mathcal{O}^A_t) \leq score(c) \leq pmax^A_t(c, \mathcal{O}^A_t)$$

Proof:

1. $score(c) \leq pmax^A_t(c, \mathcal{O}^A_t)$: By definition $score(c) = \sum_i preference^i(c)$. If c has been submitted by agent a_i ($c \in O^i_t$), then by Definition 6 $pmax^i_t(c, O^i_t) = preference^i(c)$. If $c \notin O^i_t$, $pmax^i_t(c, O^i_t) = |C| - |O^i_t| - 1$, however $c \in C \setminus O^i_t$ and thus by Borda definition $preference^i(c) \leq |C \setminus O^i_t| - 1$. Consequently, $preference^i(c) \leq pmax^i_t(c, O^i_t)$ and $\sum_i pmax^i_t(c, O^i_t) \geq \sum_i preference^i(c)$. By Definition 6 $pmax^A_t(c, \mathcal{O}^A_t) = \sum_i pmax^i_t(c, O^i_t)$, and accordingly $score(c) \leq pmax^A_t(c, \mathcal{O}^A_t)$.
2. $pmin^A_t(c, \mathcal{O}^A_t) \leq score(c)$: By definition $score(c) = \sum_i preference^i(c)$. If c has been submitted by agent a_i ($c \in O^i_t$), then by Definition 7 $pmin^i_t(c, O^i_t) = preference^i(c)$. If $c \notin O^i_t$, $pmax^i_t(c, O^i_t) = 0$, however, $c \in C \setminus O^i_t$ and thus by Borda definition $preference^i(c) \geq 0$. Consequently, $preference^i(c) \geq pmin^i_t(c, O^i_t)$ and $\sum_i pmin^i_t(c, O^i_t) \leq \sum_i preference^i(c)$. By Definition 7 $pmin^A_t(c, \mathcal{O}^A_t) = \sum_i pmin^i_t(c, O^i_t)$, and as a result $score(c) \leq pmin^A_t(c, \mathcal{O}^A_t)$. □

Now, we can prove the next theorem:

Theorem 2 *Assume C_w is the set of winners in regular Borda voting where the agents send their whole set of candidates' preferences in one iteration, then the set C_n in the final iteration t^* in iterative Borda voting affirms: $C_n = C_w$.*

Proof: The same proof as for Theorem 5, by replacing Lemma 2 with Lemma 4. □

3.3 Other voting rules

In this section we will describe variations of the iterative algorithm, for additional voting rules. However, first we note that some voting rules end in one or two iterations by definition, and thus are not a potential basis for iterative voting. For instance, in Plurality the agents submit only one candidate (the most preferred), and in Veto the agents submit solely the veto candidate (the one assigned 0). Plurality with Runoff ends in a maximum of two rounds of Plurality voting.

In this section, we focus on voting rules that usually require the whole set of candidates to determine the winner. For these rules, the basic iterative algorithm is identical to Algorithm 2. The only difference is that in the ranked-based rules the center receives the agents' next candidate ($c^i_{p_t}$, as in Borda), rather than the preference value of the candidate in ranking rules (line 9: $\langle c, d\rangle^i_t$, as in Range voting). Also, the computation of the necessary and possible winners is the same as described in Sect. 3.1. However, the rules differ in the computation of the minimum and maximum scores which are affected by the fact that the candidates are submitted in a decreasing order.

3.3.1 Rank-based rules

The next rules, in the worst case, require the whole set of agents' preferences:

Copeland: a candidate c_i receives one point if it beats candidate c_j in a majority pairwise election. The winner is the one who gathers the most points. Let us demonstrate this with Example 2, where the preferences of the agents are as follows:

$$\begin{aligned} a_1 &: c_3 > c_2 > c_1 > c_4 \\ a_2 &: c_2 > c_4 > c_3 > c_1 \\ a_3 &: c_1 > c_2 > c_3 > c_4 \end{aligned}$$

The winner in this case is c_2 with a score of 3, since it has a majority in a pairwise election against each one of the other three candidates. The score of c_1 for example is 1 since it has a majority only against c_4.

As mentioned, Algorithm 2 remains the same by letting the voters submit their candidates in a decreasing order. Algorithm 4 describes the computation of the maximum score of candidate c' based on the incomplete ordered preference values of the voters ($\mathcal{O}^A_t$). With the condition in line 7 we exploit the decreasing order by determining that if c' has already been submitted by a_i then it must beat c_j. To calculate the minimum, we omit lines 9–10, since if both c' and c_j have not been submitted by voter a_i, the pessimistic completion of the score of c' is to assume that $c' \succ c_j$.

For instance, the expected minimum and maximum of the previous example is presented in Table 5. Likewise, based on the preferences received in the first iteration, in the optimistic scenario, c_1 has a majority by beating c_2 (a_1 and a_3), and the same by beating c_3 and c_4. Thus the maximum of c_1 is 3. In the next iteration, c_2 has a majority against all the other candidates thus its minimum and maximum is 3 and it is declared as the necessary winner.

Let $score(c)$ denote the final score of c in regular Copeland voting, where the agents send their whole set of preferences in one iteration. The relation between $score(c)$ and the possible maximum and minimum values of c in an iterative Copeland voting protocol is presented in the next lemma:

Algorithm 4 CALCULATE_MAX_COPELAND
(**input**: candidate c',
ordered preference values of set A: $\mathcal{O}_t^A$
output: the max value of candidate c')

```
1: $max \leftarrow 0$
2: for all $c_j \in C$ $(c_j \neq c')$ do
3:   $inc \leftarrow 0$
4:   for all $O_t^i \in \mathcal{O}_t^A$ do
5:     if $c' \in O_t^i$ and $c_j \in O_t^i$ and $c' \succ c_j$ then
6:       $inc \leftarrow inc + 1$
7:     else if $c' \in O_t^i$ and $c_j \notin O_t^i$ then
8:       $inc \leftarrow inc + 1$
9:     else if $c' \notin O_t^i$ and $c_j \notin O_t^i$ then
10:      $inc \leftarrow inc + 1$
11:    end if
12:  end for
13:  if $inc > |A|/2$ then
14:    $max \leftarrow max + 1$
15:  end if
16: end for
17: return $max$
```

Lemma 5 *Given $\mathcal{O}_t^A$ - a decreasing order of the candidates' preferences of agents in set A sent through iteration t in iterative Copeland voting,*

$$\forall t: \ pmin_t^A(c, \mathcal{O}_t^A) \leq score(c) \leq pmax_t^A(c, \mathcal{O}_t^A)$$

Proof:

1. $score(c') \leq pmax_t^A(c', \mathcal{O}_t^A)$: In the Copeland protocol the score of c' is the number of candidates c' beats in a majority pairwise election. Therefore it is sufficient to show that if in regular Copeland, c' beats c_j then in the computation of the maximum value of c', c' must also beat c_j. c' beats c_j if in the preference order of at least half of the voters $c' \succ c_j$. Without loss of generality, assuming $c' \succ c_j$ is in the preference list of agent a_1, there are three scenarios for considering c' and c_j in the computation of the maximum value:

   (a) Both c' and c_j were submitted by a_1.
   (b) c' was submitted but c_j was not.
   (c) Both c' and c_j were not submitted by a_1.

   In all three cases Algorithm 4 gives a point to c'. Consequently, the number of points candidate c' receives in the computation of the maximum value is at least the number of points it would receive in the regular Copeland protocol, and thus $score(c') \leq pmax_t^A(c', \mathcal{O}_t^A)$.
2. $pmin_t^A(c', \mathcal{O}_t^A) \leq score(c')$: It is sufficient to show that if in regular Copeland, c' loses to c_j then in the computation of the minimum score of c', c' can not beat c_j. c' loses

Table 5 The possible maximum and minimum values of candidates $\{C_1, C_2, C_3, C_4\}$ in Copeland

t	c_1		c_2		c_3		c_4	
	max	min	max	min	max	min	max	min
1	3	0	3	0	3	0	N/A	N/A
2	2	0	3	3	2	0	2	0

to c_j if $c' \succ c_j$ is in the preference order of at most half of the voters. Without loss of generality, assuming $c_j \succ c'$ is in the preference list of agent a_1, there are three scenarios for considering c' and c_j in the computation of the minimum score:

(a) Both c' and c_j were submitted by a_1.
(b) c_j was submitted but c' was not.
(c) Both c' and c_j were not submitted by a_1.

In all these cases the minimum computation version of Algorithm 4 gives a point to c_j but not to c' and thus c' loses. Consequently, the number of points of candidate c' in the computation of the minimum score is at most the number of points it would receive in the regular Copeland protocol, and thus $pmin_t^A(c', \mathcal{O}_t^A) \leq score(c')$.

Now, we can prove the next theorem:

Theorem 3 *Assuming C_w is the set of winners in regular Copeland voting where the agents send their whole set of candidates' preferences in one iteration, then set C_n in the final iteration t^* in iterative Copeland voting affirms that $C_n = C_w$.*

Proof: The same proof as for Theorem 5, while replacing Lemma 2 with Lemma 5. □

Maximin: Let $N(c_i, c_j)$ be a function that returns the number of voters that prefer c_i to c_j, then the score of c_i is $\min_j N(c_i, c_j)$. The winner is the one with the highest score. For instance, in the previous example c_2 is the winner with score 2, since at least two agents prefer c_2 to the others. The score of c_1 on the other hand is 1 since only a_3 prefers it to the other candidates. Algorithm 5 computes the maximum score of c' with incomplete setting $\mathcal{O}_t^A$. This algorithm proceeds exactly as Algorithm 4 except lines 13–14 which check the minimality score of the pairwise comparisons. As in Copeland, to calculate the minimum we omit lines 9–10.

Algorithm 5 CALCULATE_MAX_MAXIMIN
(**input**: candidate c',
ordered preference values of set A: $\mathcal{O}_t^A$
output: the max value of candidate c')

```
1:  $max \leftarrow |C|$
2:  for all $c_j \in C$ ($c_j \neq c'$) do
3:    $inc \leftarrow 0$
4:    for all $O_t^i \in \mathcal{O}_t^A$ do
5:      if $c' \in O_t^i$ and $c_j \in O_t^i$ and $c' \succ c_j$ then
6:        $inc \leftarrow inc + 1$
7:      else if $c' \in O_t^i$ and $c_j \notin O_t^i$ then
8:        $inc \leftarrow inc + 1$
9:      else if $c' \notin O_t^i$ and $c_j \notin O_t^i$ then
10:       $inc \leftarrow inc + 1$
11:     end if
12:   end for
13:   if $inc < max$ then
14:     $max \leftarrow inc$
15:   end if
16: end for
17: return $max$
```

Based on the preferences of the agents in the previous example, the maximum of c_2 in the first iteration is 2. The reason for this is that two is the minimum number of agents that prefer c_2 to c_1 (a_1 and a_2). In iteration 2 the minimum number of agents that prefer c_2 to another

Table 6 The possible maximum and minimum values of candidates $\{C_1, C_2, C_3, C_4\}$ in Maximin

t	c_1		c_2		c_3		c_4	
	max	*min*	*max*	*min*	*max*	*min*	*max*	*min*
1	2	1	2	1	2	1	N/A	N/A
2	1	1	2	2	1	1	0	0

candidate remains two, but both for the maximum as well as for the minimum values, since in any case at least two agents prefer c_2 to any other candidate (Table 6).

Let $score(c)$ denote the final score of c in regular Maximin voting, where the agents send their whole set of preferences in one iteration. The relation between $score(c)$ and the possible maximum and minimum values of c according to an iterative Maximin voting protocol is presented in the next lemma:

Lemma 6 *Given $\mathcal{O}_t^A$ - a decreasing order of the candidates' preferences of agents in set A sent through iteration t in iterative Maximin voting,*

$$\forall t: \ pmin_t^A(c, \mathcal{O}_t^A) \le score(c) \le pmax_t^A(c, \mathcal{O}_t^A)$$

Proof:

1. $score(c') \le pmax_t^A(c', \mathcal{O}_t^A)$: In the Maximin protocol the score of c' is affected by the number of agents that prefer c' to c_j. Without loss of generality, assume $c' \succ c_j$ is in the preference list of agent a_1, and prove that in the computation of the maximum value of c', c' must win a point. There are three scenarios for considering c' and c_j in the computation of the maximum value:

   (a) Both c' and c_j were submitted by a_1.
   (b) c' was submitted but c_j was not.
   (c) Both c' and c_j were not submitted by a_1.

   In all three cases Algorithm 5 gives a point to c'. Consequently, the number of points candidate c' receives in the computation of the maximum value of c' in the iterative Maximin protocol is at least the number of points it would receive in the regular Maximin protocol, and thus $score(c') \le pmax_t^A(c', \mathcal{O}_t^A)$.
2. $pmin_t^A(c', \mathcal{O}_t^A) \le score(c')$: As in the previous proof: without loss of generality, assume $c' \succ c_j$ is in the preference list of agent a_1. In this case c' will not receive a point from this agent under the regular Maximin protocol. We prove that in the computation of the minimum value of c', c' will also not receive a point. There are three scenarios for considering c' and c_j in the computation of the minimum value:

   (a) Both c' and c_j were submitted by a_1.
   (b) c_j was submitted but c' was not.
   (c) Both c' and c_j were not submitted by a_1.

   In all three cases, the minimum computation version of Algorithm 5 gives c_j a point but does not give c' a point, thus c' loses. Consequently, the number of points candidate c' receives in the computation of the minimum score is at most the number of points it would receive under the regular Maximin protocol, and thus $pmin_t^A(c', \mathcal{O}_t^A) \le score(c')$. □

Now, we can prove the next theorem:

Theorem 4 *Assuming C_w is the set of winners in regular Maximin voting where the agents send their whole set of candidates' preferences in one iteration, then the set C_n in the final iteration t^* in iterative Maximin voting affirms that $C_n = C_w$.*

Proof: The same proof as for Theorem 5, while replacing Lemma 2 with Lemma 6. □

Bucklin: The score of candidate c_i is the smallest k in which it has a majority among the top k candidates submitted by the voters. In the previous example, c_2 is the winner with a score of 2, since revealing the top two candidates shows that c_2 has a majority. c_3, on the other hand, will have a majority only among the top three candidates. This rule could naturally proceed by submitting the candidates in a decreasing order in rounds. The first candidate(s) that has a majority is the winner. As long as a winner has not been found, the minimum and maximum scores are identical for all the candidates since either in the optimistic scenario they will have a majority in the next round or in the pessimistic scenario they will have a majority only in the last round. The correctness of the iterative Bucklin is straightforward: in regular Bucklin voting, the winner is determined by the top k candidates of the voters. The same is true in the iterative version of Bucklin where the voters reveal their preferences in a decreasing order, and thus reveal the top k preferences before the others.

In the previous example, in the first iteration no candidate has a majority, but after 2 iterations c_2 has a majority and thus it is a necessary winner.

3.3.2 Non-ranking rules

Approval: is a scoring rule which has a scoring vector $(1, 1, \ldots, 0, 0)$. Actually, it is a special case of Range voting with a domain score of $D = \{0, 1\}$.

Cumulative: A voter distributes a predefined number of points num over the candidates. This rule is different than Range, since in Cumulative the candidates' scores depend on each other since they must sum num, whereas in Range each candidate is scored independently of the others. The score of a candidate is its total score over the votes. The candidate with the highest total score is the winner. For instance, given $num = 14$, the candidates' scores are presented in Table 7 (note that the total sum of each row is 14). The winner is c_1 since it has the highest total sum.

Algorithm 6 computes the maximum optimistic score of candidate c' based on incomplete information over the candidates $\mathcal{O}_t^A$ in a Cumulative rule where the number of points to distribute over the candidates is num. This algorithm is similar to the maximum computation in Range voting (Algorithm 1) except line 14. If the center does not yet know the score of candidate c' by voter a_i, it evaluates the expected maximum of c' as the minimum between (a) the score sent by a_i in the previous round and (b) the difference between num and the summation of the points of the candidates that have already been submitted

Table 7 The candidates' preference values of the agents in set $A = \{a_1, a_2, a_3\}$

$v^i(c_j)$	c_1	c_2	c_3	c_4
a_1	5	3	3	3
a_2	2	6	3	3
a_3	7	4	2	1

by a_i $(num - sum)$. To calculate the expected minimum we replace this condition with $max \leftarrow max + max(d_1, num - sum)$.

For example, based on the scores presented in Table 7, a_1 submits 5 for c_1, a_2 submits 6 for c_2 and a_3 submits 7 for c_3. The maximum of c_1 in the first iteration is 18 since a_1 and a_3 assigned 5 and 7, respectively, and the maximum that a_2 could assign is equal to the score it already assigned c_2 (6). The minimum of c_1 is 12 since a_1 and a_3 already assigned their values (5 and 7, respectively) and the minimum that a_2 could assign c_1 is 0.

Algorithm 6 CALCULATE_MAX_CUMULATIVE
(**input**: candidate c',
ordered preference values of set A: $\mathcal{O}_t^A$,
number of points to distribute over candidates: *num*
output: the max value of candidate c')

```
1: $max \leftarrow 0$
2: for all $O_t^i \in \mathcal{O}_t^A$ do
3:   $sum \leftarrow 0$
4:   $flag \leftarrow NOT\_FOUND$
5:   for all $\langle c, d\rangle_f^i \in O_t^i$ and $flag = NOT\_FOUND$ do
6:     if $c' = c$ then
7:       $max \leftarrow max + d$
8:       $flag \leftarrow FOUND$
9:     else
10:      $sum \leftarrow sum + d$
11:    end if
12:  end for
13:  if $flag = NOT\_FOUND$ then
14:    $max \leftarrow max + min(d \in \langle c, d\rangle_t^i, num - sum)$
15:  end if
16: end for
17: return $max$
```

Let $score(c)$ denote the final score of c in regular Cumulative voting where the agents send their whole set of candidates' preference values in one iteration, then $score(c) = \sum_i v^i(c)$. The relation between $score(c)$ and the possible maximum and minimum values of c in an iterative Cumulative voting protocol is presented in the next lemma:

Lemma 7 *Given $\mathcal{O}_t^A$ - a decreasing order of the candidates' preference values of agents in set A sent through iteration t in iterative Cumulative voting,*

$$\forall t : \; pmin_t^A(c, \mathcal{O}_t^A) \leq score(c) \leq pmax_t^A(c, \mathcal{O}_t^A)$$

Proof:

1. $score(c) \leq pmax_t^A(c, \mathcal{O}_t^A)$: By definition $score(c) = \sum_i v^i(c)$. If $v^i(c)$ has been submitted by agents a_i, then by Algorithm 6 $pmax_t^i(c, O_t^i) = v^i(c)$. Otherwise, the value of $pmax_t^i(c, O_t^i)$ is either (a) $pmax_t^i(c, O_t^i) = d \in \langle c, d\rangle_t^i$, where $\langle c, d\rangle_t^i$ is the last preference value that has been sent in O_t^i, or (b) $num - sum$. In both cases $pmax_t^i(c, O_t^i) \geq v^i(c)$. Consequently, $\sum_i pmax_t^i(c, O_t^i) \geq \sum_i v^i(c)$. Based on Algorithm 6 $pmax_t^A(c, \mathcal{O}_t^A) = \sum_i pmax_t^i(c, O_t^i)$, and $score(c) \leq pmax_t^A(c, \mathcal{O}_t^A)$.
2. $pmin_t^A(c, \mathcal{O}_t^A) \leq score(c)$: If $v^i(c)$ has been submitted by agents a_i, then by the minimum computation version of Algorithm 6 $pmin_t^i(c, O_t^i) = v^i(c)$. Otherwise, $pmin_t^i(c, O_t^i) = d_1$, and thus $pmin_t^i(c, O_t^i) \leq v^i(c)$. Consequently,

$\sum_i pmin_t^i(c, O_t^i) \leq \sum_i v^i(c)$. By Algorithm 6 $pmin_t^A(c, \mathcal{O}_t^A) = \sum_i pmin_t^i(c, O_t^i)$, and $pmin_t^A(c, \mathcal{O}_t^A) \leq score(c)$. □

Now, we can prove the next theorem:

Theorem 5 *Assuming C_w is the set of winners in regular Cumulative voting where the agents send their whole set of candidates' preference values in one iteration, then the set C_n in the final iteration t^* in iterative Range voting affirms that $C_n = C_w$.*

Proof: The same proof as for Theorem 5, while replacing Lemma 2 with Lemma 7. □

To summarize, all the voting rules we have presented in this section, in the worst case require all the agents' preferences to determine the winners [3]. In the best case these rules require only one preference from each agent (for instance, if all the agents rank the same candidate first). To evaluate the average case we will present, as mentioned above, a comprehensive experimental evaluation in Sect. 5 for two representative rules: Borda and Range.

4 Greedy voting procedure

The basic idea of iterative voting (Algorithm 2) is that the center gathers information on the preferences of the agents in rounds. Increasing the amount of information (adding rounds) narrows down the possible winners set, until the procedure completes when only the necessary winners remain. Thus iterative voting as described above might still require the agents to submit a large number of preference values (i.e., through many rounds), and in the worst case the whole set [3], in order to find the necessary winners. Moreover, in some applications each such round entails interruption of a user (e.g., [14]). Thus our objectives of minimizing communication and user interruption might not be met.

To address this challenge, we now turn to a heuristic iterative voting procedure. Using the following heuristic, the center could hypothesize who is the winning candidate even with very limited information about the candidates. In this heuristic the agents send their preference values only in P pre-defined number of iterations, either in Range voting or Borda. Based on the information gathered in these P iterations the center selects the Q most preferred candidates (Q is also pre-defined). Here we evaluate the most preferred candidates as the candidates whose minimum possible scores are maximized (the minimum column in Table 8). At this point, the center asks for the completion of the preference values of only the selected Q candidates in one shot. Based on full information about the Q candidates it can compute the winner among the Q candidates by means of the pre-defined protocol. When P is unlimited this algorithm is the same as Algorithm 2.

Algorithm 7 presents the greedy heuristic for Range voting. In the main loop (lines 5–14), the agents send their candidates to the center over P iterations at most, one candidate with each iteration, in a decreasing order of preference values. Similar to Algorithm 2, in every

Table 8 The possible maximum and minimum values of candidates $\{C_1, C_2, C_3, C_4\}$

t	c_1		c_2		c_3		c_4	
	max	min	max	min	max	min	max	min
1	18	12	18	6	N/A	N/A	N/A	N/A
2	15	12	13	10	10	3	10	3
3	14	14	13	13	N/A	N/A	N/A	N/A

Algorithm 7 GREEDY_HEURISTIC
(**input**: limit number of iterations P, limit number of candidates Q
output: set of greedy winning candidates C_g)

```
1: $t \leftarrow 1$
2: $O_t^1 \leftarrow \emptyset, O_t^2 \leftarrow \emptyset, \ldots, O_t^n \leftarrow \emptyset$
3: $C_p \leftarrow \emptyset$
4: $C_n \leftarrow \emptyset$
5: while $(C_p - C_n \neq \emptyset \vee t = 1) \bigwedge j < P$ do
6:   for all $a_i \in A$ do
7:     receive $\langle c, d \rangle_t^i$
8:     $O_t^i \leftarrow O_t^i \bigcup \{\langle c, d \rangle_t^i\}$
9:   end for
10:   $\mathcal{O}_t^A \leftarrow \{O_t^1, O_t^2, \ldots, O_t^n\}$
11:   $C_p \leftarrow CALCULATE\_POSSIBLE\_WINNERS(\mathcal{O}_t^A)$
12:   $C_n \leftarrow CALCULATE\_NECESSARY\_WINNERS(\mathcal{O}_t^A)$
13:   $t \leftarrow t + 1$
14: end while
15: if $C_p - C_n = \emptyset$ then
16:   return $C_n$
17: else
18:   $C_q \leftarrow CALCULATE\_MOST\_PREFERRED\_Q(\mathcal{O}_t^A)$
19:   for all $a_i \in A$ do
20:     receive from $a_i$ values of $C_q$
21:   end for
22:   $C_w \leftarrow CALCULATE\_WINNER(C_q)$
23:   return $C_w$
24: end if
```

iteration the center calculates the possible winners (C_p) and the necessary winners (C_n) based on the minimal and maximal preference values of the candidates. If set C_p contains only necessary winners then the voting is completed (lines 15–17). Otherwise, in line 18, the algorithm invokes the CALCULATE_MOST_PREFERRED_Q Algorithm (not presented) which returns the most preferred Q candidates. It computes the possible minimal and maximal preference values of the candidates based on current partial information. Then, it sorts the candidates by the possible minimum as the first key and by the possible maximum as the second key, and returns the first Q candidates. The center asks the agents for their preference values for the Q candidates (lines 19–21) and then invokes the CALCULATE_WINNERS Algorithm (not presented) which returns the set of winners C_w out of C_q candidates, based on full information for the Q candidates.[2]

Example 3 Based on Example 1, assume $P = 2$ and $Q = 2$. After two iterations ($P = 2$) the center selects the two ($Q = 2$) candidates with the highest possible minimum as a primary key (c_1 in the second row of Table 8) and the highest possible maximum as a secondary key (c_2 has the same minimum as c_3 and c_4 but a higher maximum). The center asks the agents to complete the information about c_1 and c_2. Finally, the center selects c_1 as the winner since it has the maximal sum ($5 + 1 + 4 = 10$). Thus it chooses the optimal winner.

The greedy heuristic does not guarantee the optimal winner, since it is possible that the optimal winner is not included in the greedy Q preferred candidates after P iterations. However, as the value of P and/or Q increases, the probability that the optimal winner will be selected among the most preferred candidates also increases. Nevertheless, if P is much

[2] Although this heuristic also works for the Borda protocol, once an agent is required to complete its preference for a certain candidate it must do so in relation to the other candidates. Considering our attempt to reduce user interruption, this requirement may be much more serious in the Borda protocol than in the Range voting protocol.

Table 9 The preferences for candidates $\{c_1, c_2, c_3\}$

$v^i(c_j)$	c_1	c_2	c_3
a_1	10	1	9
a_2	1	10	9
a_3	3	2	9

smaller than the actual number of iterations required to find the optimal winner (in Algorithm 2) and/or if Q is much smaller than the size of the set of the candidates, then we expect to significantly reduce communication.

Let us demonstrate the non-optimal property of the greedy algorithm:

Example 4 Table 9 summarizes the preference values of three agents $\{a_1, a_2, a_3\}$ for candidates $\{c_1, c_2, c_3\}$ with domain $D = \{1, 2, \dots, 10\}$.

Assume $P = 1$ and $Q = 2$. After the first iteration the center selects c_1 and c_2 as the most preferred candidates, and based on the information gathered on these candidates it will select c_1 as the winner. However, unfortunately, this winner is not the optimal winner, since the total sum of c_3 is 27 in contrast to 14 of c_1. Indeed, in this case, if $P = 2$ then after two iterations the center would have selected c_1 and c_3 and would have concluded that the optimal winner is c_3.

Since the greedy algorithm does not guarantee the optimal winner, we consider the possibility of a loser candidate to win (a candidate that would have been ranked last where the voters submit their preferences in one shot). In particular, we examine the possibility of a loser to win as dependence on P and Q.

P is the number of iterations to proceed. The lower bound of P to enable a loser to be in the selected Q candidates is 1, since it suffices that at least one voter prefers the loser to include it in the Q candidates set. The upper bound of P is determined by the possibility of a loser to be in the possible winner set after P iterations, since as long as the loser exists in the possible winner set it could be included in the selected Q candidates. This possibility depends on the voting rule.

In **Range voting**, a loser could be a possible winner until the whole candidate set received by the voters. Thus, the upper bound is $P = |C|$. For example, assume n voters that assign the highest range (d_k) to all the m candidates except of one voter that assigns the lowest range (d_1) to one of the candidates. Since the voters submit their preference values in a decreasing order the loser will be proved as an impossible winner only in the last iteration. In **Borda**, an upper bound for number of voters of $n > 3$ is $P = |C|$ and for $n \leq 3$ is $P = |C| - 1$ (the proof is presented in the Appendix).

Let us demonstrate the upper bound of P on the following setting using Borda. Assume $n = 4$ voters vote over $m = 5$ candidates as presented in Table 4. The score of the loser c_1 is 7, and the score of the winner c_2 is 9. One round before the last one (after receiving the candidates with scores 4, 3 and 2), the expected minimum of c_1 is 7 and its expected maximum is 9 since the maximum next score that a_3 and a_4 can submit is 1. The minimum of the winner c_2 is 9, thus the center has to wait to the last round in order to determine that c_1 is an impossible winner (Table 10).

Now we will examine the possibility of a loser to win, depending on Q, the number of candidates that the center asks the voters about to complete the information. The question is what is the possibility of a loser to win given $|Q|$? For voting rules that satisfy the criterion of independence of irrelevant alternatives (like Range) we can guarantee that the $|Q| - 1$ losers (the $|Q|-1$ candidates that are ranked last) will not win. The reason is that the center receives the whole information about the Q candidates and no information about other candidates can influence the rank of the Q candidates. Thus, we are guaranteed that the winning candidate among the Q candidates will beat the other $Q-1$ candidates.

For voting rules that do not satisfy the criterion of independence of irrelevant alternatives (like Borda), in case that the loser exists in the possible winner set (depending on P as mentioned above) we could not guarantee that the loser will not win, since the final rank is determined only by the information about all the candidate set. Receiving the information only about part of the candidate set (Q) may determine a winner that in the full voting process is a loser. For example, assume a set of 4 voters with the following preferences:

2 voters: $c_1 > c_3 > c_4 > c_5 > c_2$
2 voters: $c_2 > c_4 > c_3 > c_5 > c_1$
1 voter: $c_5 > c_1 > c_2 > c_3 > c_4$

In this setting c_5 is the loser with 8 points. Assume $P = 1$, $Q = 3$. Then after one iteration the center will ask the voters to complete the information on c_1, c_2 and c_5. Based on the information about these candidates the center ranks them as follow: c_5 is the winner with 6 points, then c_1 with 5 points and finally c_2 with 4 points. In this example we can see that a loser can be selected as the winner.

To summarize, in Range voting a loser may exist in the possible winner set for $P \leq |C|$, but we are guaranteed that the $Q-1$ candidates ranked last in regular Range voting will not win. On the other hand, in Borda, a loser may exist in the possible winner set for $P \leq |C|$ only for a number of voters of $n > 3$, but for $n \leq 3$ a loser exists as far as $P \leq |C| - 1$. But, once it exists in the possible winner set we are not guaranteed that a loser will not win. Although this result shows that a loser may win, actually this may happen only in extreme settings where the difference between the candidates' scores is very small. Indeed, in our experiments with simulation and real-world data, the winner found by Algorithm 7 was of very high quality (see next section).

5 Evaluation

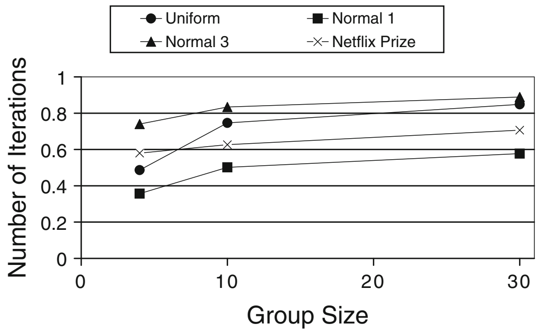
In this section we will present the evaluation of the iterative algorithm (Algorithm 2) and the greedy algorithm (Algorithm 7) in terms of communication and the quality of the selected winners. We examined these algorithms through thousands of tests in a simulation of the example of setting a meeting, and in a real world domain of movie selection using the Netflix Prize database [10].

Setting a meeting simulation: We built a simulation where every agent has a calendar of 5 days containing ten time slots a day. The agents vote for a time slot in order to set a meeting that requires all agents to participate. The agents submit their preference values for the time

Table 10 The preferences of the agents in Borda protocol, where the loser maximizes its score

$v^i(c_j)$	c_1	c_2	c_3	c_4	c_5
a_1	4	0	2	1	3
a_2	3	1	2	4	0
a_3	0	4	1	2	3
a_3	0	4	3	1	2

Fig. 1 Algorithm 2 (iterative): Number of iterations over a set size using Range voting



slots as the iterations proceed. In order to examine the influence of the distribution on the agents' preferences of the time slots, we varied our experiments according to three different distributions:

1. Uniform: the agents select the value of each time slot uniformly.
2. Normal 1: the preference values are normally distributed over the day, where the time slots in the middle of the day are assigned a higher value than the time slots in the beginning and at the end of the day.
3. Normal 3: the same as the previous distribution, but the agents are divided into three sub-sets where each one of them applies a normal distribution with a different peak.

In addition, we examined different sizes of groups to show the results for small (4 agents), medium (10 agents) and large groups (30 agents). Finally, we ran the experiments using the Borda voting protocol and the Range voting protocol. Since the preference values of the candidates were randomly assigned based on the selected distribution, we ran each experiment 50 times.

Real world movie selection applying the Netflix Prize database: Netflix provided a training data set of over 100 million ratings that over 480,000 users gave for almost 18,000 movies. The movies are ranked in a range of 1–5. We examined our algorithms through six sets each containing 50 voters for 50 movies, with a total of 300 movies and 300 people.

5.1 Evaluating the iterative algorithm

Figure 1 presents the result for the iterative algorithm (Algorithm 2) using the Range voting protocol. The x-axis represents the size of the set of voters and the y-axis represents the percentage over the maximum number of iterations (where the agents send all their candidates' preference values). Every data point in the graphs is an average of 50 test-runs, except of the Netflix curve which is an average of 6 test-runs. As proven above in Theorem 5 this algorithm finds the optimal winners. As we see in the graph our algorithm improves the worst case and much fewer iterations (number of candidates) are required in order to find the winners.

In addition, Fig. 1 shows that the number of iterations increases with the number of agents since the probability of obtaining a certain candidate's value from all the agents decreases as the group size increases (the probability that all the agents will submit their preference value for a certain candidate is $(\frac{1}{|C|})^n$, where $|C|$ is the number of candidates, and n is the number of agents). Finally, we can see that in terms of the number of iterations $Normal\,3 > Uniform > Normal\,1$. Consequently, as the homogeneity of the agents increases, i.e., their preference values for the candidates become more similar and they can find a winner faster. For instance, in $Normal\,1$ distribution, where all the agents have the

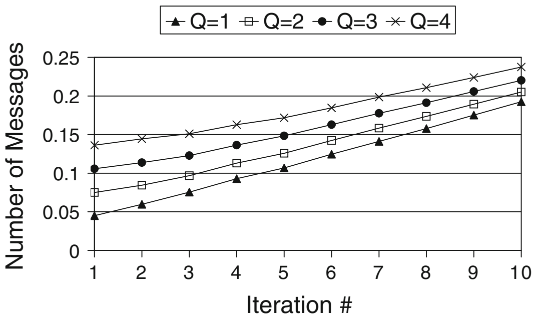


Fig. 2 Netflix Prize domain: Number of messages over iterations using Range voting in Algorithm 7 (greedy)

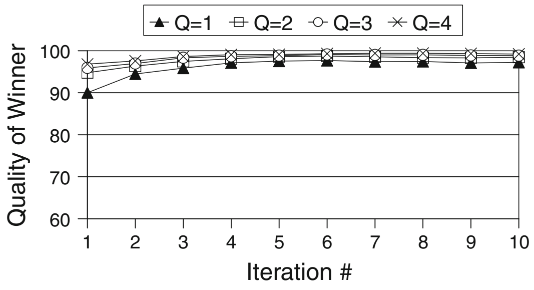


Fig. 3 Normal distribution: Quality of winner over iterations using Range voting found by Algorithm 7 (greedy)

same normal distribution and they all prefer the middle of the day rather than the edges, the agents identify the winner in the smallest number of iterations.

5.2 Evaluating the greedy algorithm

We examined the greedy algorithm (Algorithm 7) by varying $P = \{1, \dots, 10\}$ and $Q = \{1, 2, 3, 4\}$ (P is the number of iterations after which the center selects the most preferred Q candidates).

The greedy algorithm does not guarantee the optimal winner. We measured the quality of the winner it finds, by calculating its percentage rank out of the real winner computed post-hoc. In addition, we measured the number of messages needed to find the winner. The number of messages is affected by three parameters: P, Q and the set size (n). Analytically, the number of messages is bounded by $Pn + 2Qn$ (Pn for the first P iterations that the agents send and $2Qn$ for the Q candidates that the center asks about and the agents reply). However, in practice part of the Q preference values are already sent by the agents during the first P iterations, so the number of messages may be smaller.

Figure 2 presents the percentage over the maximum number of messages in the Netflix Prize (where the agents send all their candidates' preference values). The x-axis is the number of iterations (P) and the y-axis is the percentage of the max messages. As mentioned, in the Netflix domain there are at most 50 voters, but here we show the experiments in three group sizes of 4, 10 and 30. Every data point in the graph is an average of 18 test-runs (6 test-runs for every group size of voters: 4, 10, 30). As expected, as the number of iterations (P) or number of candidates to ask about (Q) increases, the number of messages also increases. Obviously, by comparing Fig. 2 to Fig. 1 we can see that the greedy algorithm saves much communication, compared to the iterative algorithm.

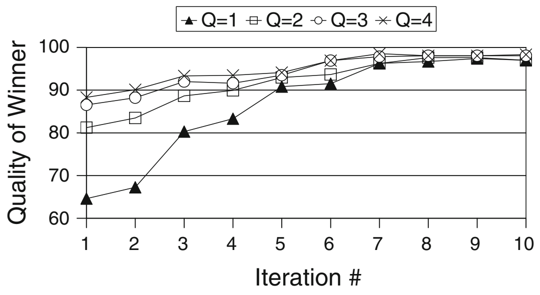


Fig. 4 Netflix Prize domain: Quality of winner over iterations using Range voting found by Algorithm 7 (greedy)

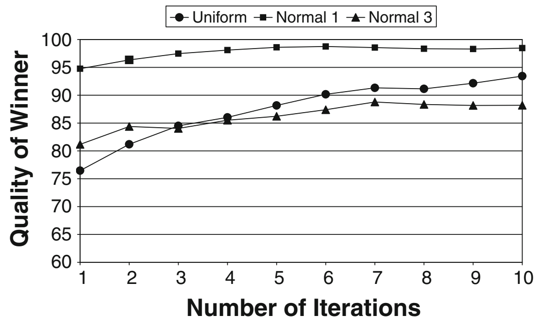


Fig. 5 Quality of the winner over iterations where $Q = 2$ in Algorithm 7 (the greedy algorithm) using Range voting

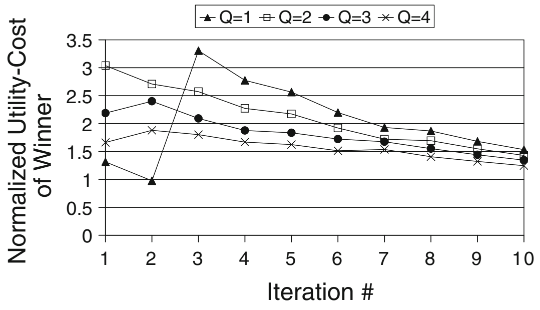


Fig. 6 Netflix Prize domain: Normalized value of the utility-cost using Range voting in Algorithm 7 (greedy). The higher the better

In Fig. 3 we examine the quality of the winner in the Range voting protocol found by the greedy algorithm (for the simulation domain of normal-1 distribution). The x-axis is the number of iterations (P) and the y-axis is the quality in percent, where 100% denotes that it finds the exact winner that would have been found in regular Range voting. In general, we can conclude that as P or Q increases, the quality of the winner increases.

In Fig. 4 we can see similar results for the Netflix Prize domain (each data point is an average of 18 tests). However, in normal distribution (Fig. 3) the quality increases more quickly than in the Netflix Prize domain, due to the homogeneity of the set, i.e., agents have similar preference values and consequently they agree more quickly about the optimal winner. To strengthen this claim we present Fig. 5 which shows the quality of the winner for three distributions (presented above) over the iterations, where $Q = 2$. All the distributions show a high quality even after a few iterations. The quality of $Normal$ 1 distribution is higher than the quality of $Uniform$ distribution and the quality of $Uniform$ distribution is higher than the quality of $Normal$ 3 distribution. This is due to the fact that the homogeneity of the agents influences the possibility of distinguishing between the candidates.

The quality of the winner is affected by the number of messages. In general the larger the number of messages the higher the quality of the winner. The question arises as to the relation between the number of messages (which is affected by P and Q) and the quality of the winner, i.e., what factor is more dominant in increasing the quality, additional iterations (P) or the additional value of more candidates (Q)? In order to examine this question in Fig. 6 we present the normalized value of the utility-cost, namely, the ratio between the number of messages and the quality of the winner based on the number of iterations (in the Netflix Prize domain). Each data point is an average of 18 tests; the higher utility-cost value the better combination of Q and P. According to this graph we can conclude, for instance, that it is preferable to wait three iterations and then ask about the value of the best two candidates than to ask about the best three candidates immediately following the first iteration. On the other hand, waiting six iterations where $Q = 2$ provides worse results (in terms of utility-cost) than one iteration where $Q = 3$. The significant increase in the curve from iteration 2 to 3 is affected by the significant increase in the quality of the winner. The best P/Q combination according to this graph is $Q = 1$ and $P = 3$.

5.3 The impact of the diversity factor

In the last set of experiments we examine the impact of the diversity of the candidates' scores on the different algorithms. We hypothesize in this section that the greater the diversity of the candidates' scores (within each of the sets of preferences of the agents), the quicker the impossible candidates will be pruned. In order to examine this hypothesis, we have simulated Borda voting by randomly extending rank-ties to non rank-ties, where ties are broken uniformly at random. For instance, in the Netflix domain, if three movie candidates where ranked with the lowest preference value 1, then in Borda ranking their rank was modified to 0, 1, 2 randomly. We ran 50 experiments while extending the order randomly. The algorithms fit Borda, where the candidates are fully distinct from each other, better than Range where the candidates are distributed over a range of only 1–5.

In Fig. 7, we can see a comparison between Borda and Range voting in terms of the **number of iterations** required to find the winners, in the iterative algorithm (Algorithm 2) in the Netflix domain. In Fig. 8 we can see a comparison between the two protocols in terms of the **quality of the winner** using the greedy algorithm (Algorithm 7) in the Netflix domain (where $Q = 2$). The improvement was achieved by increasing the diversity among the candidates' preferences in Borda, and enabling easier distinction between the candidates and thus pruning the impossible candidates faster and more accurately.

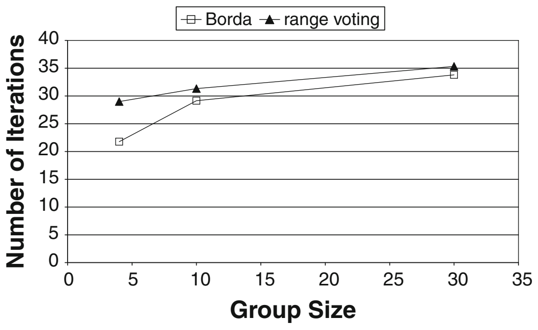


Fig. 7 A comparison between Borda voting and Range voting using Algorithm 2 in the Netflix domain

In order to strengthen our hypothesis about the influence of the diversity, we ran experiments to compare Range voting and Borda with rank-ties permitted. In Borda with rank-ties the score of a candidate is the number of candidates less than it. In order to simulate Borda with rank-ties, we changed the preference values of each agent in Range voting to a set of Borda preferences with rank-ties by keeping only the relations between the candidates but not their values. For instance, the set: $\{\langle c_1, 8\rangle \langle c_2, 5\rangle, \langle c_3, 5\rangle, \langle c_4, 1\rangle\}$ in Range voting is represented as $\{\langle c_1, 2\rangle \langle c_2, 1\rangle, \langle c_3, 1\rangle, \langle c_4, 0\rangle\}$ in Borda with rank-ties.

Generally, the algorithms fit Range voting better than Borda since the candidates' diversity in Range setting is greater than that of Borda with rank-ties setting. For instance, in Fig. 9, we can see a comparison between Borda with rank-ties and Range voting in terms of the **number of iterations** required to find the winners, in the iterative algorithm (Algorithm 2) with $Normal$ 1 distribution. In Fig. 10 we can see a comparison between the two protocols in terms of the **quality of the winner** in the greedy algorithm (Algorithm 7) with $Normal$ 3 distribution. The reason for this improvement is that in Range voting the calculation of the maximum and minimum of the candidates takes into consideration the preference values of the candidates and not only their order as in Borda with rank-ties. Thus, Range voting increases the diversity among the candidates' preference values, and enables easier distinction between the candidates for pruning the impossible candidates.

The last experiments that show the effect of the diversity compare Borda with no rank-ties to Borda with rank-ties in the Netflix domain. We attained the same results which show that regular Borda, where the candidates set is more diverse, outperforms Borda with rank-ties (Figs. 11 and 12).

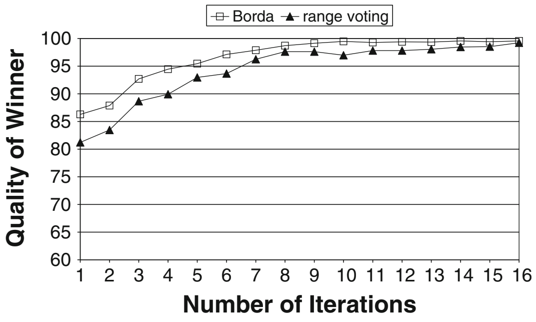


Fig. 8 A comparison between Borda and Range voting using Algorithm 7 (the greedy algorithm) in the Netflix domain

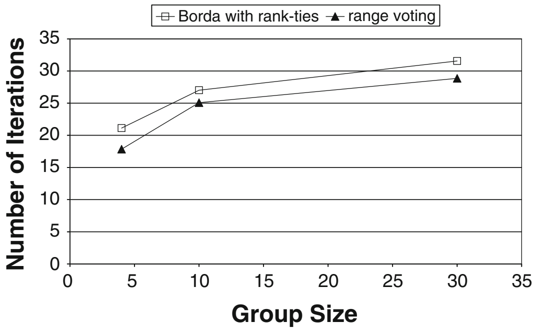


Fig. 9 A comparison between Borda voting and Range voting using Algorithm 2 for $Normal$ 1 distribution

Fig. 10 A comparison between Borda and Range voting using Algorithm 7 (the greedy algorithm) for *Normal* 3 distribution

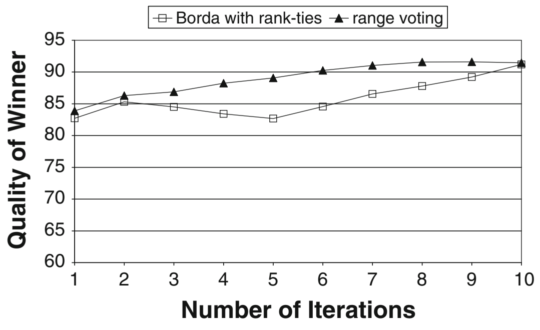


Fig. 11 A comparison between Borda and Borda with rank-ties using Algorithm 2 in the Netflix domain

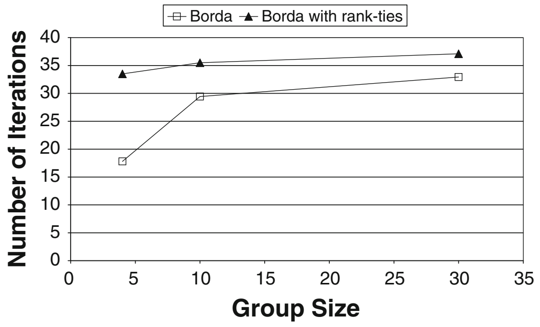
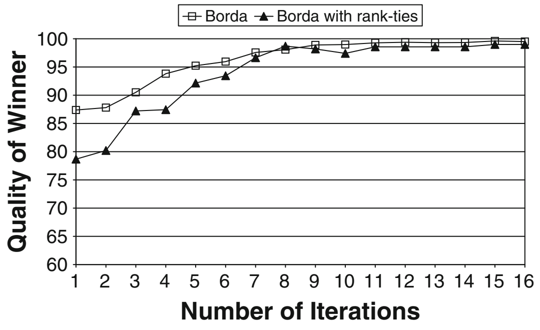


Fig. 12 A comparison between Borda and Borda with rank-ties using Algorithm 7 (the greedy algorithm) in the Netflix domain



6 Summary

Voting protocols typically assume that all agents provide their entire set of preferences over candidates to the center agent which collects the votes and determines the winner. However, in practice, this assumption may entail significant costs (e.g., in communication bandwidth) and effort (e.g., requiring users to enter their preferences needlessly).

In this paper we presented an iterative voting protocol which addresses this problem. The key to iterative voting is that agents submit their preferences for candidates incrementally, in rounds. We described the implementation of this protocol in Range voting and in Borda. We presented several variants of iterative voting (for different voting rules), and proved their completeness. We then presented a greedy heuristic algorithm to further reduce communication, and show that while it is not guaranteed to find the true winner, it significantly reduces communication while keeping the high quality of the winner.

We have empirically shown that: (a) the iterative algorithm reduces communication in relation to a regular protocol in which agents submit their entire information; (b) as the set size decreases and the distribution of the preference values over the agents is homogeneous, the number of preference values that are required for the voting decreases. For the greedy voting algorithm we have demonstrated that in environments with low bandwidth or where communication is expensive, we can significantly reduce communication costs while trading off a small loss in quality. In addition, we have established that the homogeneity also influences the quality of the winner in the greedy voting algorithm. Finally, we have empirically showed that increasing the diversity of the candidates enables easier pruning of the impossible candidates.

In the future we plan to improve the greedy algorithm. Currently, we set the number of iterations (P) and the number of best candidates (Q) in advance. However, these parameters depend on the number of agents, the number of candidates, the distribution of the candidates' preference values etc. We plan to investigate the effect of these factors on P and Q, and consequently attempt to determine these parameters online. In addition, in this paper we focused on voting protocols in which the agents submit a single preference value in rounds. It may be challenge to investigate compact representation in iterative voting where the preference of the candidates are result of multi-criteria calculation.

Appendix

The upper bound of P (the number of iterations to proceed in Algorithm 7) is determined by the possibility of a loser to be in the possible winner set after P iterations.

Theorem 6 *In Borda, an upper bound for a group size of* $n > 3$ *is* $P = |C|$ *and for* $n \leq 3$ *is* $P = |C| - 1$.

Proof Let us assume an extreme voting case where the loser has the highest possible score: $\frac{(m-1)n}{2} - 1$. In this case the gap between the winner and the loser must be 2. The votes, in this case, that maximize the expected maximum of the loser is: $(n/2) - 1$ voters rank the loser first, one voter ranks it second $(m - 2)$ and $n/2$ of the voters rank it last. In this vote setting, after two rounds the expected minimum score of the loser is its real score which is less in 2 points from that of the winner. The expected maximum of the loser at that point is its current score $((\frac{n}{2} - 1)(m - 1) + m - 2)$ plus the possible maximum score received by $n/2$ of the voters that have not submitted yet their score for the loser. In order to prune the loser from the possible winners set, the expected maximum of the loser must be less than the expected minimum of the winner. Specifically, **one round before the last one** the maximum score that an agent could assign a candidate that has not submitted yet is 1. Thus, the additional expected maximum for those agents that have not submitted yet their preference is $\frac{n}{2} * 1$, and the condition for the loser to be pruned from the possible winners set is:

$$\left(\frac{n}{2} - 1\right)(m - 1) + m - 2 + \frac{n}{2} < \left(\frac{n}{2} - 1\right)(m - 1) + m$$

by investigating this inequality we conclude that the condition is $n < 4$. Meaning, in a group of $n \leq 3$ voters, a loser is not able to be a possible winner after $|C| - 1$ iterations. For group size of $n > 3$ this inequality does not hold so the minimum of the winner could not be greater than the maximum of the loser and so it could not be pruned in the round before the last but in the last round.

On the other hand, the maximum score that an agent could assign a candidate that has not submitted yet **two rounds before the last one** is 2. Thus, the condition for the loser to be pruned from the possible winners set is:

$$\left(\frac{n}{2}-1\right)(m-1)+m-2+\frac{n}{2}\cdot 2<\left(\frac{n}{2}-1\right)(m-1)+m$$

This inequality entails $n < 2$, but a voting system is defined for $n \geq 2$. The constant in the right side of the inequality $n < 2$ monotonically decreases as the number of rounds before the last one increases. Thus, we conclude, based on the expected maximum in the round before the last round, that the upper bound for $n > 3$ is $P = |C|$ and for $n \leq 3$ is $P = |C| - 1$. $\square$

Acknowledgements We thank Israeli Ministry of Science grant number 3-6797 and ISF Grant number 1357/07. As always, thanks to K. Ushi and K. Ravit.

References

1. Blosser, G., & Zhan, J. (2008). Privacy-preserving collaborative e-voting. In *Intelligence and security informatics* (pp. 508–513). Berlin, Heidelberg: Springer.
2. Conitzer, V., & Sandholm, T. (2002). Vote elicitation: Complexity and strategy-proofness. In *Eighteenth national conference on artificial intelligence*, pp. 392–397.
3. Conitzer, V., & Sandholm, T. (2005). Communication complexity of common voting rules. In *EC '05: Proceedings of the 6th ACM conference on electronic commerce*, pp. 78–87.
4. Faltings, B., & Macho-Gonzalez, S. (2005). Open constraint programming. *Artifitial Intelligence, 161*(1–2), 181–208.
5. Gelain, M., Pini, M. S., Rossi, F., & Venable, K. B. (2007) Dealing with incomplete preferences in soft constraint problems. In *The 13th international conference on principles and practice of constraint prgramming (CP-07)*, pp. 286–300.
6. Hudson, B., & Sandholm, T. (2003). Generalizing preference elicitation in combinatorial auctions. In *Proceedings of the second international joint conference on autonomous agents and multiagent systems (AAMAS)*, pp. 1014–1015.
7. Konczak, K. (2006). Voting theory in answer set programming. In *The 20th workshop on logic programming (WLP-06)*, pp. 45–53.
8. Konczak, K., & Lang, J. (2005). Voting procedures with incomplete preferences. In *Proceedings of multidisciplinary IJCAI'05 workshop on advances in preference handling*, Edinburg, Scotland.
9. Lang, J., Pini, M. S., Rossi, F., Venable, K. B., & Walsh, T. (2007). Winner determination in sequential majority voting. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)*, pp. 1372–1377.
10. Netflix prize: http://www.netflixprize.com.
11. Pini, M. S., Rossi, F., Venable, K. B., & Walsh, T. (2007). Incompleteness and incomparability in preference aggregation. In M. M. Veloso (Ed.), *In Proceedings of the international joint conference on artificial intelligence (IJCAI)*, pp. 1464–1469.
12. Pini, M. S., Rossi, F., Venable, K. B., & Walsh, T. (2009). Aggregating partially ordered preferences. *Journal of Logic and Computation, 19*(3), 475–502.
13. Pitt, J., Kamara, L., Sergot, M., & Artikis, A. (2006). Voting in multi-agent systems. *Computer Journal, 49*(2), 156–170.
14. Tambe, M., Bowring, E., Pearce, J. P., Varakantham, P., Scerri, P., & Pynadath, D. V. (2006). Electric elves: What went wrong and why. In *Proceedings of the AAAI spring symposium on what went wrong and why: Lessons from AI research and applications*.
15. Walsh, T. (2007). Uncertainty in preference elicitation and aggregation. In *Proceedings of the 22nd national conference on Artificial intelligence (AAAI-07)*, pp. 3–8.
16. Walsh, T. (2008). Complexity of terminating preference elicitation. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pp. 967–974.

17. Xia, L., & Conitzer, V. (2008) Determining possible and necessary winners under common voting rules given partial orders. In *Proceedings of the 23rd national conference on artificial intelligence (AAAI-08)*, pp. 196–201.
18. Zhan, J., Matwin, S., & Chang, L.-W. (2004). Privacy-preserving electronic voting. *Information & Security: An International Journal, 15*(2), 165–180.