Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Approximation and Parameterized Complexity of Minimax Approval Voting

Marek Cygan
University of Warsaw, Poland
cygan@mimuw.edu.pl

Łukasz Kowalik
University of Warsaw, Poland
kowalik@mimuw.edu.pl

Arkadiusz Socała
University of Warsaw, Poland
arkadiusz.socala@mimuw.edu.pl

Krzysztof Sornat
University of Wrocław, Poland
krzysztof.sornat@cs.uni.wroc.pl

Abstract

We present three results on the complexity of MINIMAX APPROVAL VOTING. First, we study MINIMAX APPROVAL VOTING parameterized by the Hamming distance d from the solution to the votes. We show MINIMAX APPROVAL VOTING admits no algorithm running in time $\mathcal{O}^\star(2^{o(d \log d)})$, unless the Exponential Time Hypothesis (ETH) fails. This means that the $\mathcal{O}^\star(d^{2d})$ algorithm of Misra et al. [AAMAS 2015] is essentially optimal. Motivated by this, we then show a parameterized approximation scheme, running in time $\mathcal{O}^\star((3/\epsilon)^{2d})$, which is essentially tight assuming ETH. Finally, we get a new polynomial-time randomized approximation scheme for MINIMAX APPROVAL VOTING, which runs in time $n^{\mathcal{O}(1/\epsilon^2 \cdot \log(1/\epsilon))} \cdot \mathrm{poly}(m)$, almost matching the running time of the fastest known PTAS for CLOSEST STRING due to Ma and Sun [SIAM J. Comp. 2009].

1 Introduction

One of the central problems in artificial intelligence and computational social choice is aggregating preferences of individual agents (see the overview of Conitzer (2010)). Here we focus on *multi-winner choice*, where the goal is to select a k-element subset of a set of candidates. Given preferences of the agents over the candidates, a multi-winner voting rule can be used to select a subset of candidates that in some sense are preferred by the agents. This scenario covers a variety of settings: nations elect members of parliament or societies elect committees (Chamberlin and Courant 1983), web search engines choose pages to display in response to a query (Dwork et al. 2001), airlines select movies available on board (Skowron, Faliszewski, and Lang 2015), companies select a group of products to promote (Lu and Boutilier 2011), etc.

In this work we restrict our attention to approval-based multi-winner rules, i.e., rules where each voter expresses his or her preferences by providing a *subset of the candidates* which he or she approves. Various voting rules are studied in the literature. In the simplest one, Approval Voting (AV), occurrences of each candidate are counted and k most often approved candidates are selected. While this rule has many desirable properties in the single winner case (Fishburn 1978), in the multi-winner scenario its merits are often considered less clear (Laslier and Sanver 2010), e.g., because it fails to reflect the diversity of interests in the electorate (Kilgour 2010). Therefore, numerous alternative rules have been proposed, including Satisfaction Approval Voting, Proportional Approval Voting, and Reweighted Approval Voting (see Kilgour (2010) for details). In this paper we study a rule called Minimax Approval Voting (MAV), introduced by Brams, Kilgour, and Sanver (2007). Here, we see the votes and the choice as 0-1 strings of length m (characteristic vectors of the subsets, i.e., the candidate i is approved if the string contains 1 at position i). For two strings x and y of the same length the Hamming distance $\mathcal{H}(x, y)$ is the number of positions where x and y differ, e.g., $\mathcal{H}(011, 101) = 2$. In MAV, we look for a 0-1 string with k ones that minimizes the maximum Hamming distance to a vote. In other words, MAV minimizes the disagreement with the least satisfied voter and thus it is highly egalitarian: no voter is ignored and a majority of voters cannot guarantee a specific outcome (LeGrand 2004; Brams, Kilgour, and Sanver 2007).

Our focus is on the computational complexity of computing the choice based on the MAV rule. In the MINIMAX APPROVAL VOTING decision problem, we are given a multiset $S = \{s_1, \ldots, s_n\}$ of 0-1 strings of length m (also called votes), and two integers k and d. The question is whether there exists a string $s \in \{0, 1\}^m$ with exactly k ones such that for every $i = 1, \ldots, n$ we have $\mathcal{H}(s, s_i) \leq d$. In the optimization version of MINIMAX APPROVAL VOTING we minimize d, i.e., given a multiset S and an integer k as before, the goal is to find a string $s \in \{0, 1\}^m$ with exactly k ones which minimizes $\max_{i=1,\ldots,n} \mathcal{H}(s, s_i)$.

A reader familiar with string problems might recognize that MINIMAX APPROVAL VOTING is tightly connected with the classical NP-complete problem called CLOSEST STRING, where we are given n strings over an alphabet Σ and the goal is to find a string that minimizes the maximum Hamming distance to the given strings. Indeed, LeGrand, Markakis, and Mehta (2007) showed that MINIMAX APPROVAL VOTING is NP-complete as well by reduction from CLOSEST STRING with binary alphabet. First proof of NP-completeness of MINIMAX APPROVAL VOTING was shown using reduction from VERTEX COVER (LeGrand 2004). This motivated the study

on MINIMAX APPROVAL VOTING in terms of approximability and fixed-parameter tractability.

Previous results on MINIMAX APPROVAL VOTING First approximation result was a simple 3-approximation algorithm due to LeGrand, Markakis, and Mehta (2007), obtained by choosing an arbitrary vote and taking any k approved candidates from the vote (extending it arbitrarily to k candidates if needed). Next, a 2-approximation was shown by Caragiannis, Kalaitzis, and Markakis (2010) using an LP-rounding procedure. Finally, recently Byrka and Sornat (2014) presented a polynomial time approximation scheme (PTAS), i.e., an algorithm that for any fixed $\epsilon > 0$ gives a $(1+\epsilon)$-approximate solution in polynomial time. More precisely, their algorithm runs in time $m^{\mathcal{O}(1/\epsilon^4)} + n^{\mathcal{O}(1/\epsilon^3)}$ which is polynomial in the number of voters n and the number of alternatives m. The PTAS uses information extraction techniques from fixed size ($\mathcal{O}(1/\epsilon)$) subsets of voters and random rounding of the optimal solution of a linear program.

In the area of fixed parameter tractability (FPT) every instance I of a problem P contains additionally an integer r, called a *parameter*. The goal is to find a *fixed parameter algorithm* (also called FPT algorithm), i.e., an algorithm with running time of the form $f(r)\mathrm{poly}(|I|)$, where f is a function, which is typically at least exponential for NP-complete problems. If such an algorithm exists, we say that the problem P parameterized by r is fixed parameter tractable (FPT). For more details about FPT algorithms see the textbook of Cygan et al. (2015) or the survey of Bredereck et al. (2014) (in the context of computational social choice). The study of FPT algorithms for MINIMAX APPROVAL VOTING was initiated by Misra, Nabeel, and Singh (2015). They show for example that MINIMAX APPROVAL VOTING parameterized by k (the number of ones in the solution) is $W[2]$-hard, which implies that there is no FPT algorithm, unless there is a highly unexpected collapse in parameterized complexity classes. From a positive perspective, they show that the problem is FPT when parameterized by the maximum allowed distance d or by the number of votes n. Their algorithm runs in time[1] $\mathcal{O}^\star(d^{2d})$.[2] For a study on FPT complexity of generalizations of MINIMAX APPROVAL VOTING see Baumeister et al. (2016).

Previous results on CLOSEST STRING It is interesting to compare the known results on MINIMAX APPROVAL VOTING with the corresponding ones on the better researched CLOSEST STRING. The first PTAS for CLOSEST STRING was given by Li, Ma, and Wang (2002) with running time bounded by $n^{\mathcal{O}(1/\epsilon^4)}$ where n is the number of the input strings. This was later improved by Andoni, Indyk, and Patrascu (2006) to $n^{\mathcal{O}(\frac{\log 1/\epsilon}{\epsilon^2})}$, and then by Ma and Sun (2009) to $n^{\mathcal{O}(1/\epsilon^2)}$.

The first FPT algorithm for CLOSEST STRING, running in time $\mathcal{O}^\star(d^d)$ was given by Gramm, Niedermeier, and Rossmanith (2003). This was later improved by Ma and Sun (2009), who gave an algorithm with running time $\mathcal{O}^\star(2^{\mathcal{O}(d)} \cdot |\Sigma|^d)$, which is more efficient for constant-size alphabets. Further substantial progress is unlikely, since Lokshtanov, Marx, and Saurabh (2011b) have shown that CLOSEST STRING admits no algorithms running in time $\mathcal{O}^\star(2^{o(d \log d)})$ or $\mathcal{O}^\star(2^{o(d \log |\Sigma|)})$, unless the Exponential Time Hypothesis (ETH) (Impagliazzo and Paturi 2001) fails.

The discrepancy between the state of the art for CLOSEST STRING and MINIMAX APPROVAL VOTING raises interesting questions. First, does the additional constraint on the number of ones in MINIMAX APPROVAL VOTING really make the problem harder and the PTAS has to be significantly slower? Similarly, although in MINIMAX APPROVAL VOTING the alphabet is binary, no $\mathcal{O}^\star(2^{\mathcal{O}(d)})$-time algorithm is known, in contrast to CLOSEST STRING. Can we find such an algorithm? The goal of this work is to answer these questions.

Our results We present three results on the complexity of MINIMAX APPROVAL VOTING. Let us recall that the Exponential Time Hypothesis (ETH) of Impagliazzo and Paturi (2001) states that there exists a constant $c > 0$, such that there is no algorithm solving 3-SAT in time $\mathcal{O}^\star(2^{cn})$. In recent years, ETH became the central conjecture used for proving tight bounds on the complexity of various problems, see Lokshtanov, Marx, and Saurabh (2011a) for a survey. Nevertheless, ETH-based lower bounds seem largely unexplored in the area of computational social choice (Niedermeier 2015). We begin with showing that, unless the ETH fails, there is no algorithm for MINIMAX APPROVAL VOTING running in time $\mathcal{O}^\star(2^{o(d \log d)})$. In other words, the algorithm of Misra, Nabeel, and Singh (2015) is essentially optimal, and indeed, in this sense MINIMAX APPROVAL VOTING is harder than CLOSEST STRING. Motivated by this, we then show a parameterized approximation scheme, i.e., a randomized Monte-Carlo algorithm which, given an instance (S, k, d) and a number $\epsilon > 0$, finds a solution at distance at most $(1+\epsilon)d$ in time $\mathcal{O}^\star((3/\epsilon)^{2d})$ or reports that there is no solution at distance at most d (with arbitrarily small positive constant probability of error). Note that our lower bound implies that, under (randomized version of) ETH, this is essentially optimal, i.e., there is no parameterized approximation scheme running in time $\mathcal{O}^\star(2^{o(d \log(1/\epsilon))})$. Indeed, if such an algorithm existed, by picking $\epsilon = 1/(d+1)$ we would get an exact algorithm which contradicts our lower bound. Finally, we get a new polynomial-time randomized approximation scheme for MINIMAX APPROVAL VOTING, which runs in time $n^{\mathcal{O}(1/\epsilon^2 \cdot \log(1/\epsilon))} \cdot \mathrm{poly}(m)$ (with arbitrarily small positive constant probability of error). Thus the running time almost matches the one of the fastest known PTAS for CLOSEST STRING (up to a $\log(1/\epsilon)$ factor in the

[1] The $\mathcal{O}^\star$ notation suppresses factors polynomial in the input size.

[2] Actually, in the article (Misra, Nabeel, and Singh 2015) the authors claim the slightly better running time of $\mathcal{O}^\star(d^d)$. However, there is a flaw in the analysis (Misra 2016; Liu and Guo 2016): it states that the initial solution v is at distance at most d from the solution, while it can be at distance $2d$ because of what we call here the k-completion operation. This increases the maximum depth of the recursion to d (instead of the claimed $d/2$).

exponent).

Organization of the paper In Section 2 we introduce some notation and we recall standard probability bounds that are used later in the paper. In Section 3 we present our lower bound for MINIMAX APPROVAL VOTING parameterized by d. Next, in Section 4 we show a parameterized approximation scheme. Finally, in Section 5 we show a new randomized PTAS. The paper concludes with Section 6, where we discuss directions for future work.

2 Definitions and Preliminaries

For every integer n we denote $[n] = \{1, 2, \ldots, n\}$. For a set of words $S \subseteq \{0,1\}^m$ and a word $x \in \{0,1\}^m$ we denote $\mathcal{H}(x, S) = \max_{s \in S} \mathcal{H}(x, s)$. For a string $s \in \{0,1\}^m$, the number of 1's in s is denoted as $n_1(s)$ and it is also called the Hamming weight of s; similarly $n_0(s) = m - n_1(s)$ denotes the number of zeroes. Moreover, the set of all strings of length m with k ones is denoted by $S_{k,m}$, i.e., $S_{k,m} = \{s \in \{0,1\}^m : n_1(s) = k\}$. $s[j]$ means j-th letter of a string s. For a subset of positions $P \subseteq [m]$ we define a subsequence $s|_P$ by removing the letters on positions $[m] \setminus P$ from s.

For a string $s \in \{0,1\}^m$, any string $s' \in S_{k,m}$ at distance $|n_1(s) - k|$ from s is called a *k-completion* of s. Note that it is easy to find such a k-completion s': when $n_1(s) \geq k$ we obtain s' by replacing arbitrary $n_1(s) - k$ ones in s by zeroes; similarly when $n_1(s) < k$ we obtain s' by replacing arbitrary $k - n_1(s)$ zeroes in s by ones.

3 A lower bound

In this section we show a lower bound for MINIMAX APPROVAL VOTING parameterized by d. To this end, we use a reduction from a problem called $k \times k$-CLIQUE. In $k \times k$-CLIQUE we are given a graph G over the vertex set $V = [k] \times [k]$, i.e., V forms a grid (as a vertex set; the edge set of G is a part of the input and it can be arbitrary) with k rows and k columns, and the question is whether in G there is a clique containing exactly one vertex in each row.

Lemma 3.1. *Given an instance $I = (G, k)$ of $k \times k$-CLIQUE with $k \geq 2$, one can construct an instance $I' = (S, k, d)$ of MINIMAX APPROVAL VOTING, such that I' is a yes-instance iff I is a yes-instance, $d = 3k - 3$ and the set S contains $\mathcal{O}(k\binom{2k-2}{k-2})$ strings of length $k^2 + 2k - 2$ each. The construction takes time polynomial in the size of the output.*

Proof. Each string in the set S will be of size $m = k^2 + 2k - 2$. Let us split the set of positions $[m]$ into $k+1$ blocks, where the first k blocks contain exactly k positions each, and the last $(k+1)$-th block contains the remaining $2k-2$ positions. Our construction will enforce that if a solution exists, it will have the following structure: there will be a single 1 in each of the first k blocks and only zeroes in the last block. Intuitively the position of the 1 in the first block encodes the clique vertex of the first row of G, the position of the 1 in the second block encodes the clique vertex of the second row of G, etc.

We construct the set S as follows.

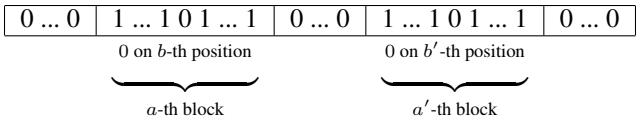

Figure 1: Nonedge string.

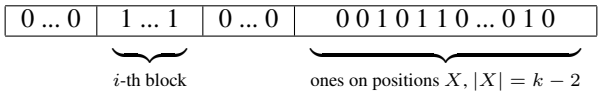

Figure 2: Row string.

- **(nonedge strings)** For each pair of nonadjacent vertices $v, v' \in V(G)$ of G belonging to different rows, i.e., $v = (a, b)$, $v' = (a', b')$, $a \neq a'$, we add to S a string $s_{vv'}$, where all the blocks except a-th and a'-th are filled with zeroes, while the blocks a, a' are filled with ones, except the b-th position in block a and the b'-th position in block a' which are zeroes (see Fig. 1). Formally, $s_{vv'}$ contains ones at positions $\{(a-1)k + j : j \in [k], j \neq b\} \cup \{(a'-1)k + j : j \in [k], j \neq b'\}$. Note that the Hamming weight of $s_{vv'}$ equals $2k - 2$.
- **(row strings)** For each row $i \in [k]$ we create exactly $\binom{2k-2}{k-2}$ strings, i.e., for $i \in [k]$ and for each set X of exactly $k-2$ positions in the $(k+1)$-th block we add to S a string $s_{i,X}$ having ones at all positions of the i-th block and at X, all the remaining positions are filled with zeroes (see Fig. 2). Note that similarly as for the nonedge strings the Hamming weight of each row string equals $2k-2$, and to achieve this property we use the $(k+1)$-th block.

To finish the description of the created instance $I' = (S, k, d)$ we need to define the target distance d, which we set as $d = 3k - 3$. Observe that as the Hamming weight of each string $s' \in S$ equals $2k - 2$, for $s \in \{0,1\}^m$ with exactly k ones we have $\mathcal{H}(s, s') \leq d$ if and only if the positions of ones in s and s' have a non-empty intersection.

Let us assume that there is a clique K in G of size k containing exactly one vertex from each row. For $i \in [k]$ let $j_i \in [k]$ be the column number of the vertex of K from row i. Define s as a string containing ones exactly at positions $\{(i-1)k + j_i : i \in [k]\}$, i.e., the $(k+1)$-th block contains only zeroes and for $i \in [k]$ the i-th block contains a single 1 at position j_i. Obviously s contains exactly k ones, hence it suffices to show that s has at least one common one with each of the strings in S. This is clear for the row strings, as each row string contains a block full of ones. For a nonedge string $s_{vv'}$, where $v = (a, b)$ and $v' = (a', b')$ note that K does not contain v and v' at the same time. Consequently s has a common one with $s_{vv'}$ in at least one of the blocks a, a'.

In the other direction, assume that s is a string of length m with exactly k ones such that the Hamming distance between s and each of the strings in S is at most d, which by construction implies that s has a common one with each of the strings in S. First, we are going to prove that s contains a 1 in each of the first k blocks (and consequently has

only zeroes in block $k+1$). For the sake of contradiction assume that this is not the case. Consider a block $i \in [k]$ containing only zeroes. Let X be any set of $k-2$ positions in block $k+1$ holding only zeroes in s (such a set exists as block $k+1$ has $2k-2$ positions). But the row string $s_{i,X}$ has $2k-2$ ones at positions where s has zeroes, and consequently $\mathcal{H}(s, s_{i,X}) = k + (2k-2) = 3k-2 > d = 3k-3$, a contradiction.

As we know that s contains exactly one one in each of the first k blocks let $j_i \in [k]$ be such a position of block $i \in [k]$. Create $X \subseteq V(G)$ by taking the vertex from column j_i for each row $i \in [k]$. Clearly X is of size k and it contains exactly one vertex from each row, hence it remains to prove that X is a clique in G. Assume the contrary and let $v, v' \in X$ be two distinct nonadjacent vertices of X, where $v = (i, j_i)$ and $v' = (i', j_{i'})$. Observe that the nonedge string $s_{vv'}$ contains zeroes at the j_i-th position of the i-th block and at the $j_{i'}$-th position of the i'-th block. Since for $i'' \in [k]$, $i'' \neq i$, $i'' \neq i'$ block i'' of $s_{vv'}$ contains only zeroes, we infer that the sets of positions of ones of s and $s_{vv'}$ are disjoint leading to $\mathcal{H}(s, s_{vv'}) = k + (2k-2) = 3k-2 > d$, a contradiction.

As we have proved that I is a yes-instance of $k \times k$-CLIQUE iff I' is a yes-instance of MINIMAX APPROVAL VOTING, the lemma follows. □

In order to derive an ETH-based lower bound we need the following theorem of Lokshtanov, Marx, and Saurabh (2011b).

Theorem 3.2. *(Lokshtanov, Marx, and Saurabh 2011b) Assuming ETH, there is no $2^{o(k \log k)}$-time algorithm for $k \times k$-*CLIQUE.

We are ready to prove the main result of this section.

Theorem 3.3. *There is no $2^{o(d \log d)}\mathrm{poly}(n, m)$-time algorithm for* MINIMAX APPROVAL VOTING *unless ETH fails.*

Proof. Using Lemma 3.1, the input instance G of $k \times k$-CLIQUE is transformed into an equivalent instance $I' = (S, k, d)$ of MINIMAX APPROVAL VOTING, where $n = |S| = \mathcal{O}(k\binom{2k-2}{k-2}) = 2^{\mathcal{O}(k)}$, each string of S has length $m = \mathcal{O}(k^2)$ and $d = \Theta(k)$. Using a $2^{o(d \log d)}\mathrm{poly}(n, m)$-time algorithm for MINIMAX APPROVAL VOTING we can solve $k \times k$-CLIQUE in time $2^{o(k \log k)}2^{\mathcal{O}(k)} = 2^{o(k \log k)}$, which contradicts ETH by Theorem 3.2. □

4 Parameterized approximation scheme

In this section we show the following theorem.

Theorem 4.1. *There exists a randomized algorithm which, given an instance $(\{s_i\}_{i=1,\ldots,n}, k, d)$ of* MINIMAX APPROVAL VOTING *and any $\epsilon \in (0, 3)$, runs in time $\mathcal{O}\left(\left(\frac{3}{\epsilon}\right)^{2d} mn\right)$ and either*

- (i) *reports a solution at distance at most $(1+\epsilon)d$ from S, or*
- (ii) *reports that there is no solution at distance at most d from S.*

In the latter case, the answer is correct with probability at least $1 - p$, for arbitrarily small fixed $p > 0$.

Pseudocode 1: Parameterized approximation scheme for MINIMAX APPROVAL VOTING.

```
if |n_1(s_1) − k| > d then return NO;
x_0 ← any k-completion of s_1;
for j ∈ {1, 2, ..., d} do
    if H(x_{j−1}, S) ≤ (1 + ε)d then return x_{j−1};
    otherwise there exists s_i s.t. H(x_{j−1}, s_i) > (1 + ε)d;
    P_{j,0} ← {a ∈ [m] : 0 = x_{j−1}[a] ≠ s_i[a] = 1};
    P_{j,1} ← {a ∈ [m] : 1 = x_{j−1}[a] ≠ s_i[a] = 0};
    if min(|P_{j,0}|, |P_{j,1}|) = 0 then return NO;
    Get x_j from x_{j−1} by swapping 0 and 1 on pair of random
      positions from P_{j,0} and P_{j,1});
if H(x_d, S) ≤ (1 + ε)d then return x_d;
else return NO ;
```

Let us proceed with the proof. In what follows we assume $p = 1/2$, since then we can get the claim even if $p < 1/2$ by repeating the whole algorithm $\lceil \log_2(1/p) \rceil$ times. Indeed, then the algorithm returns an incorrect answer only if each of the $\lceil \log_2(1/p) \rceil$ repetitions returned an incorrect answer, which happens with probability at most $(1/2)^{\log_2(1/p)} = p$.

Assume we are given a yes-instance and let us fix a solution $s^* \in S_{k,m}$, i.e., a string at distance at most d from all the input strings. Our approach is to begin with a string $x_0 \in S_{k,m}$ not very far from s^*, and next perform a number of steps. In the j-th step we either conclude that x_{j-1} is already a $(1+\epsilon)$-approximate solution, or with some probability we find another string x_j which is closer to s^*.

First observe that if $|n_1(s_1) - k| > d$, then clearly there is no solution and our algorithm reports NO. Hence in what follows we assume

$$|n_1(s_1) - k| \leq d. \tag{1}$$

We set x_0 to be any k-completion of s_1. By (1) we get $\mathcal{H}(x_0, s_1) \leq d$. Since $\mathcal{H}(s_1, s^*) \leq d$, by the triangle inequality we get the following bound.

$$\mathcal{H}(x_0, s^*) \leq \mathcal{H}(x_0, s_1) + \mathcal{H}(s_1, s^*) \leq 2d. \tag{2}$$

Now we are ready to describe our algorithm precisely (see also Pseudocode 1). We begin with x_0 defined as above. We are going to create a sequence of strings $x_0, x_1, \ldots$ satisfying $n_1(x_j) = k$ for every j. For $j = 1, \ldots, d$ we do the following. If for every $i = 1, \ldots, n$ we have $\mathcal{H}(x_{j-1}, s_i) \leq (1+\epsilon)d$ the algorithm terminates and returns x_{j-1}. Otherwise, fix any $i = 1, \ldots, n$ such that $\mathcal{H}(x_{j-1}, s_i) > (1+\epsilon)d$. Let $P_{j,0} = \{a \in [m] : 0 = x_{j-1}[a] \neq s_i[a] = 1\}$ and $P_{j,1} = \{a \in [m] : 1 = x_{j-1}[a] \neq s_i[a] = 0\}$. The algorithm samples a position $a_0 \in P_{j,0}$ and a position $a_1 \in P_{j,1}$. In case $P_{j,0} = \emptyset$ or $P_{j,1} = \emptyset$ we return NO because it means that $\mathcal{H}(s_i, S_{k,m}) = \mathcal{H}(s_i, x_{j-1}) > d$. Then, x_j is obtained from x_{j-1} by swapping the 0 at position a_0 with the 1 at position a_1. If the algorithm finishes without finding a solution, it reports NO.

The following lemma is the key to get a lower bound on the probability that the x_j's get close to s^*.

Lemma 4.2. *Let x be a string in $S_{k,m}$ such that $\mathcal{H}(x, s_i) \geq (1+\epsilon)d$ for some $i = 1, \ldots, n$. Let $s^* \in S_{k,m}$ be any solution,*

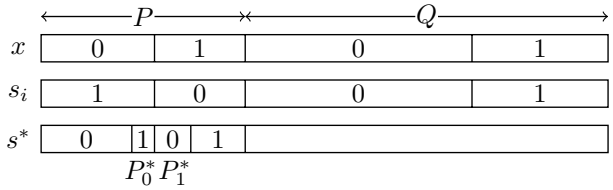

Figure 3: Strings x, s_i and s^* after permuting the positions.

i.e., a string at distance at most d from all the strings s_i, $i = 1, \ldots, n$. Denote

$$P_0^* = \{a \in [m] : 0 = x[a] \neq s_i[a] = s^*[a] = 1\},$$
$$P_1^* = \{a \in [m] : 1 = x[a] \neq s_i[a] = s^*[a] = 0\}.$$

Then,

$$\min(|P_0^*|, |P_1^*|) \geq \frac{\epsilon d}{2}.$$

Proof. Let P be the set of positions on which x and s_i differ, i.e., $P = \{a \in [m] : x[a] \neq s_i[a]\}$ (see Fig. 3). Note that $P_0^* \cup P_1^* \subseteq P$. Let $Q = [m] \setminus P$.

The intuition behind the proof is that if $\min(|P_0^*|, |P_1^*|)$ is small, then s^* differs too much from s_i, either because $s^*|_P$ is similar to $x|_P$ (when $|P_0^*| \approx |P_1^*|$) or because $s^*|_Q$ has much more 1's than $s_i|_Q$ (when $|P_0^*|$ differs much from $|P_1^*|$).

We begin with a couple of useful observations on the number of ones in different parts of x, s_i and s^*. Since x and s_i are the same on Q, we get

$$n_1(x|_Q) = n_1(s_i|_Q). \tag{3}$$

Since $n_1(x) = n_1(s^*)$, we get $n_1(x|_P) + n_1(x|_Q) = n_1(s^*|_P) + n_1(s^*|_Q)$, and further

$$n_1(s^*|_Q) - n_1(x|_Q) = n_1(x|_P) - n_1(s^*|_P). \tag{4}$$

Finally note that

$$n_1(s^*|_P) = |P_0^*| + n_1(x|_P) - |P_1^*|. \tag{5}$$

We are going to derive a lower bound on $\mathcal{H}(s_i, s^*)$. First,

$$\mathcal{H}(s_i|_P, s^*|_P) = |P| - (|P_0^*| + |P_1^*|) =$$
$$= \mathcal{H}(x, s_i) - (|P_0^*| + |P_1^*|) \geq (1+\epsilon)d - (|P_0^*| + |P_1^*|).$$

On the other hand,

$$\begin{aligned}\mathcal{H}(s_i|_Q, s^*|_Q) &\geq |n_1(s^*|_Q) - n_1(s_i|_Q)| = \\ &\overset{(3)}{=} |n_1(s^*|_Q) - n_1(x|_Q)| = \\ &\overset{(4)}{=} |n_1(x|_P) - n_1(s^*|_P)| = \\ &\overset{(5)}{=} ||P_1^*| - |P_0^*||.\end{aligned}$$

It follows that

$$\begin{aligned}d \geq \mathcal{H}(s_i, s^*) &= \mathcal{H}(s_i|_P, s^*|_P) + \mathcal{H}(s_i|_Q, s^*|_Q) \geq \\ &\geq (1+\epsilon)d - (|P_0^*| + |P_1^*|) + ||P_1^*| - |P_0^*|| = \\ &= (1+\epsilon)d - 2\min(|P_0^*|, |P_1^*|).\end{aligned}$$

Hence, $\min(|P_0^*|, |P_1^*|) \geq \frac{\epsilon d}{2}$ as required. □

Corollary 4.3. *Assume that there is a solution $s^* \in S_{k,m}$ and that the algorithm created a string x_j, for some $j = 0, \ldots, d$. Then,*

$$\Pr[\mathcal{H}(x_j, s^*) \leq 2d - 2j] \geq \left(\frac{\epsilon}{3}\right)^{2j}.$$

Proof. We use induction on j. For $j = 0$ the claim follows from (2). Consider $j > 0$. By the induction hypothesis,

$$\Pr[\mathcal{H}(x_{j-1}, s^*) \leq 2d - 2j + 2] \geq \left(\frac{\epsilon}{3}\right)^{2j-2}. \tag{6}$$

Assume that $\mathcal{H}(x_{j-1}, s^*) \leq 2d - 2j + 2$. Since x_j was created, $\mathcal{H}(x_{j-1}, s_i) > (1+\epsilon)d$ for some $i = 1, \ldots, n$. Since $\mathcal{H}(s^*, s_i) \leq d$, by the triangle inequality we get the following.

$$\begin{aligned}&|P_{j,0}| + |P_{j,1}| = \mathcal{H}(x_{j-1}, s_i) \leq \\ &\leq \mathcal{H}(x_{j-1}, s^*) + \mathcal{H}(s^*, s_i) \leq 3d - 2j + 2 \leq 3d.\end{aligned} \tag{7}$$

Then, by Lemma 4.2

$$\begin{aligned}\Pr[\mathcal{H}(x_j, s^*) \leq 2d - 2j \mid \mathcal{H}(x_{j-1}, s^*) \leq 2d - 2j + 2] \geq \\ \geq \frac{|P_0^*| \cdot |P_1^*|}{|P_{j,0}| \cdot |P_{j,1}|} \geq \frac{\left(\frac{\epsilon d}{2}\right)^2}{\left(\frac{3d}{2}\right)^2} = \left(\frac{\epsilon}{3}\right)^2.\end{aligned} \tag{8}$$

The claim follows by combining (6) and (8). □

In order to increase the success probability, we repeat the algorithm until a solution is found or the number of repetitions is at least $(3/\epsilon)^{2d}$. By Corollary 4.3 the probability that there is a solution but it was not found is bounded by

$$\left(1 - \left(\frac{\epsilon}{3}\right)^{2d}\right)^{(3/\epsilon)^{2d}} = \left(1 - \frac{1}{(3/\epsilon)^{2d}}\right)^{(3/\epsilon)^{2d}} \leq \frac{1}{e} < \frac{1}{2}.$$

This finishes the proof of Theorem 4.1.

5 A faster polynomial time approximation scheme

The goal of this section is to present a PTAS for the optimization version of MINIMAX APPROVAL VOTING running in time $n^{\mathcal{O}(1/\epsilon^2 \cdot \log(1/\epsilon))} \cdot \mathrm{poly}(m)$. It is achieved by combining the parameterized approximation scheme from Theorem 4.1 with the following result, which might be of independent interest. Throughout this section OPT denotes the value of an optimum solution s for the given instance $(\{s_i\}_{i=1,\ldots,n}, k)$ of MINIMAX APPROVAL VOTING, i.e., $\mathrm{OPT} = \max_{i=1,\ldots,n} \mathcal{H}(s, s_i)$,

Theorem 5.1. *There exists a randomized polynomial time algorithm which, for arbitrarily small fixed $p > 0$, given an instance $(\{s_i\}_{i=1,\ldots,n}, k)$ of* MINIMAX APPROVAL VOTING *and any $\epsilon > 0$ such that $\mathrm{OPT} \geq \frac{122 \ln n}{\epsilon^2}$, reports a solution, which with probability at least $1 - p$ is at distance at most $(1+\epsilon) \cdot \mathrm{OPT}$ from S.*

In what follows, we prove Theorem 5.1. As in the proof of Theorem 4.1 we assume w.l.o.g. $p = 1/2$. Note that we can assume $\epsilon < 1$, for otherwise it suffices to

Pseudocode 2: The algorithm from Theorem 5.1

```
Solve the LP (9–12) obtaining an optimal solution (x*_1, ..., x*_m, d*);
for j ∈ {1, 2, ..., m} do
    Set x[j] ← 1 with probability x*_j and x[j] ← 0 with probability 1 − x*_j
y ← any k-completion of x;
return y
```

use the 2-approximation of Caragiannis, Kalaitzis, and Markakis (2010). We also assume $n \geq 3$, for otherwise it is a straightforward exercise to find an optimal solution in linear time. Let us define a linear program (9–12):

$$\text{minimize} \quad d \tag{9}$$

$$\sum_{j=1}^{m} x_j = k \tag{10}$$

$$\sum_{\substack{j=1,\ldots,m \\ s_i[j]=1}} (1 - x_j) + \sum_{\substack{j=1,\ldots,m \\ s_i[j]=0}} x_j \leq d \qquad \forall i \in \{1, \ldots, n\} \tag{11}$$

$$x_j \in [0, 1] \qquad \forall j \in \{1, \ldots, m\} \tag{12}$$

The linear program (9–12) is a relaxation of the natural integer program for MINIMAX APPROVAL VOTING, obtained by replacing (12) by the discrete constraint $x_j \in \{0, 1\}$. Indeed, observe that x_j corresponds to the j-th letter of the solution $x = x_1 \cdots x_m$, (10) states that $n_1(x) = k$, and (11) states that $\mathcal{H}(x, S) \leq d$.

Our algorithm is as follows (see Pseudocode 2). First we solve the linear program in time $\text{poly}(n, m)$ using the interior point method (Karmarkar 1984). Let $(x_1^*, \ldots, x_m^*, d^*)$ be the obtained optimal solution. Clearly, $d^* \leq \text{OPT}$. We randomly construct a string $x \in \{0, 1\}^m$, guided by the values x_j^*. More precisely, for every $j = 1, \ldots, m$ independently, we set $x[j] = 1$ with probability x_j^*. Note that x needs not contain k ones. Let y by any k-completion of x. The algorithm returns y.

Clearly, the above algorithm runs in polynomial time. In what follows we bound the probability of error. To this end we prove upper bounds on the probability that x is far from S and the probability that the number of ones in x is far from k. This is done in Lemmas 5.2 and 5.3, which can be shown using standard Chernoff bounds (due to space limitations, the proofs are omitted, and can be found in the full version of the paper (Cygan et al. 2016a)).

Lemma 5.2.

$$\Pr\left[\mathcal{H}(x, S) > (1 + \tfrac{\epsilon}{2}) \cdot \text{OPT}\right] \leq \tfrac{1}{4}.$$

Lemma 5.3.

$$\Pr\left[|n_1(x) - k| > \tfrac{\epsilon}{2} \cdot \text{OPT}\right] \leq \tfrac{1}{4}.$$

Now we can finish the proof of Theorem 5.1. By Lemmas 5.2 and 5.3 with probability at least $1/2$ both $\mathcal{H}(x, S) \leq (1 + \frac{1}{2}\epsilon) \cdot \text{OPT}$ and $\mathcal{H}(y, x) = |n_1(x) - k| \leq \frac{1}{2}\epsilon \cdot \text{OPT}$. By the triangle inequality this implies that $\mathcal{H}(y, S) \leq (1 + \epsilon) \cdot \text{OPT}$, with probability at least $1/2$ as required.

We conclude the section by combining Theorems 4.1 and 5.1 to get a fast PTAS.

Theorem 5.4. *For each $\epsilon > 0$ we can find $(1 + \epsilon)$-approximation solution for the* MINIMAX APPROVAL VOTING *problem in time $n^{\mathcal{O}(\frac{\log 1/\epsilon}{\epsilon^2})} \cdot \text{poly}(m)$ with probability at least $1 - r$, for any fixed $r > 0$.*

Proof. First we run algorithm from Theorem 4.1 for $d = \lceil \frac{122 \ln n}{\epsilon^2} \rceil$ and $p = r/2$.

If it reports a solution, for every $d' \leq d$ we apply Theorem 4.1 with $p = r/2$ and we return the best solution. If $\text{OPT} \geq d$, even the initial solution is at distance at most $(1 + \epsilon)d \leq (1 + \epsilon)\text{OPT}$ from S. Otherwise, at some point $d' = \text{OPT}$ and we get $(1+\epsilon)$-approximation with probability at least $1 - r/2 > 1 - r$.

In the case when the initial run of the algorithm from Theorem 4.1 reports NO, we just apply the algorithm from Theorem 5.1, again with $p = r/2$. With probability at least $1 - r/2$ the answer NO of the algorithm from Theorem 4.1 is correct. Conditioned on that, we know that $\text{OPT} > d \geq \frac{122 \ln n}{\epsilon^2}$ and then the algorithm from Theorem 5.1 returns a $(1+\epsilon)$-approximation with probability at least $1 - r/2$. Thus, the answer is correct with probability at least $(1 - r/2)^2 > 1 - r$.

The total running time can be bounded as follows.

$$\mathcal{O}^*\left(\left(\frac{3}{\epsilon}\right)^{\frac{244 \ln n}{\epsilon^2}}\right) \subseteq n^{\mathcal{O}(\frac{\log 1/\epsilon}{\epsilon^2})} \cdot \text{poly}(m).$$

□

6 Further research

We conclude the paper with some questions related to this work that are left unanswered. Our PTAS for MINIMAX APPROVAL VOTING is randomized, and it seems there is no direct way of derandomizing it. It might be interesting to find an equally fast deterministic PTAS. The second question is whether there are even faster PTASes for CLOSEST STRING or MINIMAX APPROVAL VOTING. Recently, Cygan et al. (2016b) showed that under ETH, there is no PTAS in time $f(\epsilon) \cdot n^{o(1/\epsilon)}$ for CLOSEST STRING. This extends to the same lower bound for MINIMAX APPROVAL VOTING, since we can try all values $k = 0, 1, \ldots, m$. It is a challenging open problem to close the gap in the running time of PTAS either for CLOSEST STRING or for MINIMAX APPROVAL VOTING.

Acknowledgments. Marek Cygan would like to thank Daniel Lokshtanov for helpful conversations about existing algorithms for the Closest (Sub)String problem. The authors thank Piotr Skowron for helpful remarks concerning the introduction and they thank reviewers for their insightful comments on the paper. The work of M. Cygan is a part of the project TOTAL that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 677651). Ł. Kowalik and A. Socała were supported by the National Science Centre, Poland,

grant number 2013/09/B/ST6/03136. K. Sornat was supported by the National Science Centre, Poland, grant number 2015/17/N/ST6/03684.

References

Andoni, A.; Indyk, P.; and Patrascu, M. 2006. On the Optimality of the Dimensionality Reduction Method. In *47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006*, 449–458.

Baumeister, D.; Böhnlein, T.; Rey, L.; Schaudt, O.; and Selker, A. 2016. Minisum and Minimax Committee Election Rules for General Preference Types. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 1656–1657. IOS Press.

Brams, S. J.; Kilgour, D. M.; and Sanver, M. R. 2007. A Minimax Procedure for Electing Committees. *Public Choice* 132(3-4):401–420.

Bredereck, R.; Chen, J.; Faliszewski, P.; Guo, J.; Niedermeier, R.; and Woeginger, G. J. 2014. Parameterized Algorithmics for Computational Social Choice: Nine Research Challenges. *Tsinghua Science and Technology* 19(4):358–373.

Byrka, J., and Sornat, K. 2014. PTAS for Minimax Approval Voting. In *Proceedings of 10th International Conference Web and Internet Economics, WINE 2014*, 203–217.

Caragiannis, I.; Kalaitzis, D.; and Markakis, E. 2010. Approximation Algorithms and Mechanism Design for Minimax Approval Voting. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI*.

Chamberlin, J. R., and Courant, P. N. 1983. Representative Deliberations and Representative Decisions: Proportional Representation and the Borda Rule. *American Political Science Review* 77:718–733.

Conitzer, V. 2010. Making Decisions Based on the Preferences of Multiple Agents. *Commun. ACM* 53(3):84–94.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Cygan, M.; Kowalik, Ł.; Socała, A.; and Sornat, K. 2016a. Approximation and Parameterized Complexity of Minimax Approval Voting. *CoRR* abs/1607.07906.

Cygan, M.; Lokshtanov, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2016b. Lower Bounds for Approximation Schemes for Closest String. In *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016*, 12:1–12:10.

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001. Rank Aggregation Methods for the Web. In *Proceedings of the Tenth International World Wide Web Conference, WWW 2001*, 613–622.

Fishburn, P. C. 1978. Axioms for Approval Voting: Direct Proof. *Journal of Economic Theory* 19(1):180–185.

Gramm, J.; Niedermeier, R.; and Rossmanith, P. 2003. Fixed-Parameter Algorithms for Closest String and Related Problems. *Algorithmica* 37(1):25–42.

Impagliazzo, R., and Paturi, R. 2001. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62(2):367–375.

Karmarkar, N. 1984. A New Polynomial-time Algorithm for Linear Programming. *Combinatorica* 4(4):373–396.

Kilgour, D. M. 2010. Approval Balloting for Multi-winner Elections. In Laslier, J.-F., and Sanver, R. M., eds., *Handbook on Approval Voting*. Berlin, Heidelberg: Springer Berlin Heidelberg. 105–124.

Laslier, J., and Sanver, M. 2010. *Handbook on Approval Voting*. Studies in Choice and Welfare. Springer Berlin Heidelberg.

LeGrand, R.; Markakis, E.; and Mehta, A. 2007. Some Results on Approximating the Minimax Solution in Approval Voting. In *6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2007*, 1193–1195.

LeGrand, R. 2004. Analysis of the Minimax Procedure. Technical Report WUCSE-2004-67, Department of Computer Science and Engineering, Washington University, St. Louis, Missouri.

Li, M.; Ma, B.; and Wang, L. 2002. On the Closest String and Substring Problems. *Journal of the ACM* 49(2):157–171.

Liu, H., and Guo, J. 2016. Parameterized Complexity of Winner Determination in Minimax Committee Elections. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2016*, 341–349.

Lokshtanov, D.; Marx, D.; and Saurabh, S. 2011a. Lower Bounds Based on the Exponential Time Hypothesis. *Bulletin of the EATCS* 105:41–72.

Lokshtanov, D.; Marx, D.; and Saurabh, S. 2011b. Slightly Superexponential Parameterized Problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, 760–776.

Lu, T., and Boutilier, C. 2011. Budgeted Social Choice: From Consensus to Personalized Decision Making. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, 280–286.

Ma, B., and Sun, X. 2009. More Efficient Algorithms for Closest String and Substring Problems. *SIAM Journal of Computing* 39(4):1432–1443.

Misra, N.; Nabeel, A.; and Singh, H. 2015. On the Parameterized Complexity of Minimax Approval Voting. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015*, 97–105.

Misra, N. 2016. personal communication.

Niedermeier, R. 2015. Lower Bound Issues in Computational Social Choice. A talk at the workshop Satisfiability Lower Bounds and Tight Results for Parameterized and Exponential-Time Algorithms, Simons Institute, Berkeley, November 10, 2015.

Skowron, P. K.; Faliszewski, P.; and Lang, J. 2015. Finding a Collective Set of Items: From Proportional Multirepresentation to Group Recommendation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2015*, 2131–2137.