# Approximation and Parameterized Complexity of Minimax Approval Voting

**Marek Cygan**        CYGAN@MIMUW.EDU.PL
**Łukasz Kowalik**        KOWALIK@MIMUW.EDU.PL
**Arkadiusz Socała**        ARKADIUSZ.SOCALA@MIMUW.EDU.PL
*Department of Mathematics*
*Informatics and Mechanics*
*University of Warsaw, Poland*

**Krzysztof Sornat**        KRZYSZTOF.SORNAT@CS.UNI.WROC.PL
*Department of Mathematics and Computer Science*
*University of Wrocław, Poland*

## 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, Nabeel and Singh 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 $n^{\mathcal{O}(1/\epsilon^2 \cdot \log(1/\epsilon))} \cdot \text{poly}(m)$, where $n$ is a number of voters and $m$ is a number of alternatives. It almost matches the running time of the fastest known PTAS for CLOSEST STRING due to Ma and Sun.

## 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 election*, 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 & Courant, 1983), web search engines choose pages to display in response to a query (Dwork, Kumar, Naor, & Sivakumar, 2001), airlines select movies available on board (Skowron, Faliszewski, & Lang, 2016; Elkind, Faliszewski, Skowron, & Slinko, 2017), companies select a group of products to promote (Lu & 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 & 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 (for details see a book chapter of Kilgour, 2010).

In this paper we study a rule called Minimax Approval Voting (MAV), introduced by Brams, Kilgour, and Sanver (2007b). 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, & Sanver, 2007a). John Rawls (1971) in his classical book "A Theory of Justice" states that welfare is maximized when the utility of those society members that have the least is the greatest, so MAV is a *Rawlsian social welfare function*. For more general studies on minisum (utilitarian) and minimax (egalitarian) objectives see the work of Brams et al. (2007a). Other egalitarian voting rules have been also studied by Betzler, Slinko, and Uhlmann (2013).

Much recent research has been devoted to the axiomatic properties of multi-winner voting rules (Elkind et al., 2017; Faliszewski, Skowron, Slinko, & Talmon, 2017; Sanchez-Fernandez & Fisteus, 2017). The goal is to classify and describe properties of voting rules because different voting rules have different properties, thus they should be used in proper scenarios. Let us look at some properties of MAV. MAV is not a proportional type of voting rule. A large group of voters with similar preferences can be not represented by the number of committee members proportional to the size of the group. Formally, it was shown that MAV does not satisfy *Justified Representation* property (Aziz et al., 2017b). MAV is not strategy-proof (Caragiannis, Kalaitzis, & Markakis, 2010), i.e., voters can vote strategically to be less dissatisfied by the outcome. On the other hand MAV *supports strong monotonicity with population increase* and *weak monotonicity without population increase* (Sanchez-Fernandez & Fisteus, 2017), and it is insensitive to clones (Kilgour, 2010).

MAV could be used when, for example, proportionality is not needed, agents are not selfish, but the outcome is required to be acceptable by every voter. A natural such scenario is when a group of experts have to make a decision or a group of friends want to choose activities for a holiday. Brams et al. (2007a) "commend the minimax procedure to colleges, universities, and other organizations" (p. 403).

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 $i \in \{1, \ldots, n\}$ 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 $\max_{i \in \{1,\ldots,n\}} \mathcal{H}(s, s_i)$.

A reader familiar with string problems might recognize that MINIMAX APPROVAL VOTING is closely related to the classical NP-complete problem called CLOSEST STRING, where we are given $n$ strings over an alphabet $\Sigma$ 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. The 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.

## 1.1 Previous Results on Minimax Approval Voting

The first approximation result was a simple 3-approximation algorithm due to LeGrand et al. (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 et al. (2010) using an LP-rounding procedure. Finally, 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 $n^{\mathcal{O}(1/\epsilon^4)} \cdot m^{\mathcal{O}(1)} + n^{\mathcal{O}(1/\epsilon)} \cdot m^{\mathcal{O}(1/\epsilon^4)}$ 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 to 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 $f(r)\text{poly}(|I|)$, where $f$ is a computable 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 et al. (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 it does not admit an 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 the work of Baumeister, Bohnlein, Rey, Schaudt, and Selker (2016).

## 1.2 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 $n^{\mathcal{O}(1/\epsilon^4)}$

---

1. The $\mathcal{O}^\star$ notation suppresses factors polynomial in the input size.
2. Actually, Misra et al. (2015) claim the slightly better running time of $\mathcal{O}^\star(d^d)$. However, there is a flaw in the analysis (Liu & Guo, 2016; Misra, 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 $d/2$).

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 & 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.

## 1.3 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 $\mathcal{O}^\star(2^{cn})$, where $n$ is the number of variables in the given 3-SAT instance. In recent years, ETH became the central conjecture used for proving tight bounds on the complexity of various problems, see the survey of Lokshtanov, Marx, and Saurabh (2011a). 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 et al. (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 $n^{\mathcal{O}(1/\epsilon^2 \cdot \log(1/\epsilon))} \cdot \text{poly}(m)$ (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 exponent). The new PTAS is much faster than the previous one (Byrka & Sornat, 2014). In particular, the new running time does not contain the $m^{\mathcal{O}(1/\epsilon^4)}$ term, so one should expect a considerable speed-up when the number of votes is large.

### 1.4 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 using randomized and approximation algorithms in multi-winner elections and 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., $S_{k,m} = \{s \in \{0, 1\}^m : n_1(s) = k\}$. $s[j]$ for $j \in [m]$ means the $j$-th letter of a string $s$. For a subset of positions $P \subseteq [m]$ we define a subsequence $s|_P$ by removing the letters at 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. Some readers may recognize $k \times k$-Clique as a special case of Multicolored Clique, where we have $k$ colors and every color has size exactly $k$.

**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 $\mathcal{O}\big(k\binom{2k-2}{k-2}\big)$ 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 length $m = k^2 + 2k - 2$. 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.

- **(nonedge strings)** For each pair of nonadjacent vertices $v, v' \in V$ 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 Figure 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 Figure 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.

| $0 \ldots 0$ | $1 \ldots 1 \, 0 \, 1 \ldots 1$ | $0 \ldots 0$ | $1 \ldots 1 \, 0 \, 1 \ldots 1$ | $0 \ldots 0$ |
|---|---|---|---|---|
| | 0 on $b$-th position | | 0 on $b'$-th position | |
| | $a$-th block | | $a'$-th block | |

Figure 1: Nonedge string.

| $0 \ldots 0$ | $1 \ldots 1$ | $0 \ldots 0$ | $0 \, 0 \, 1 \, 0 \, 1 \, 1 \, 0 \ldots 0 \, 1 \, 0$ |
|---|---|---|---|
| | $i$-th block | | ones on positions $X, |X| = k - 2$ |
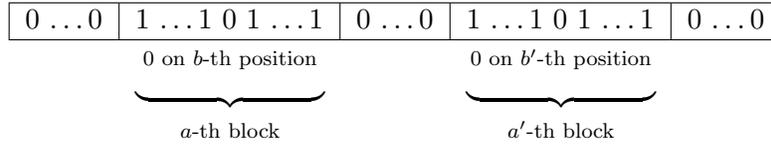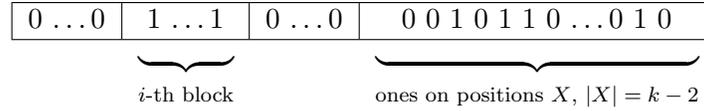
Figure 2: Row string.

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 $X \subseteq V$ 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. $\square$

In order to derive an ETH-based lower bound we need the following theorem of Lokshtanov et al. (2011b).

**Theorem 3.2.** *(Lokshtanov et al., 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. $\square$

Since this section presents a conditional lower bound, its main practical value is better understanding of the hardness of breaking the bound (now proven to be equivalent to disproving ETH). The result can be also seen as an evidence of limitations of any heuristic for MAV used in practice.

## 4. Parameterized Approximation Scheme

In this section we show the following theorem.

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

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 $|n_1(s_1) - k| \le d$. We set $x_0$ to be any $k$-completion of $s_1$, therefore we get $\mathcal{H}(x_0, s_1) \le d$. Since $\mathcal{H}(s_1, s^*) \le d$, by the triangle inequality we get the following bound

$$\mathcal{H}(x_0, s^*) \le \mathcal{H}(x_0, s_1) + \mathcal{H}(s_1, s^*) \le 2d. \tag{1}$$

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 $x_0, x_1, \ldots, x_d$ satisfying $n_1(x_j) = k$ for every $j$. For $j \in [d]$ we do the following. If for every $i \in [n]$ we have $\mathcal{H}(x_{j-1}, s_i) \le (1 + \epsilon)d$ the algorithm terminates and returns $x_{j-1}$. Otherwise, fix any $i \in [n]$ such that $\mathcal{H}(x_{j-1}, s_i) > (1 + \epsilon)d$. Let $P_{j,0} = \{a \in [m] : 0 = x_{j-1}[a] \ne s_i[a] = 1\}$ and $P_{j,1} = \{a \in [m] : 1 = x_{j-1}[a] \ne s_i[a] = 0\}$. 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.

---

**Pseudocode 1:** Parameterized approximation scheme for MINIMAX APPROVAL VOTING.

---

1   **if** $|n_1(s_1) - k| > d$ **then return** NO;
2   $x_0 \leftarrow$ any $k$-completion of $s_1$;
3   **for** $j \in \{1, 2, \ldots, d\}$ **do**
4      **if** $\mathcal{H}(x_{j-1}, S) \le (1 + \epsilon)d$ **then return** $x_{j-1}$;
5      otherwise there exists $s_i$ s.t. $\mathcal{H}(x_{j-1}, s_i) > (1 + \epsilon)d$;
6      $P_{j,0} \leftarrow \{a \in [m] : 0 = x_{j-1}[a] \ne s_i[a] = 1\}$;
7      $P_{j,1} \leftarrow \{a \in [m] : 1 = x_{j-1}[a] \ne s_i[a] = 0\}$;
8      **if** $\min(|P_{j,0}|, |P_{j,1}|) = 0$ **then return** NO;
9      Get $x_j$ from $x_{j-1}$ by swapping 0 and 1 on pair of random positions from $P_{j,0}$ and $P_{j,1}$;
10   **if** $\mathcal{H}(x_d, S) \le (1 + \epsilon)d$ **then return** $x_d$;
11   **else return** 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 $i \in [n]$. Let $s^* \in S_{k,m}$ be any solution, i.e., a string at distance at most $d$ from all the strings $s_j$, $j \in [n]$. 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, it holds that $\min(|P_0^*|, |P_1^*|) \geq \frac{\epsilon d}{2}$.*

*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 Figure 3). Note that $P_0^* \cup P_1^* \subseteq P$. Let $Q = [m] \setminus P$.
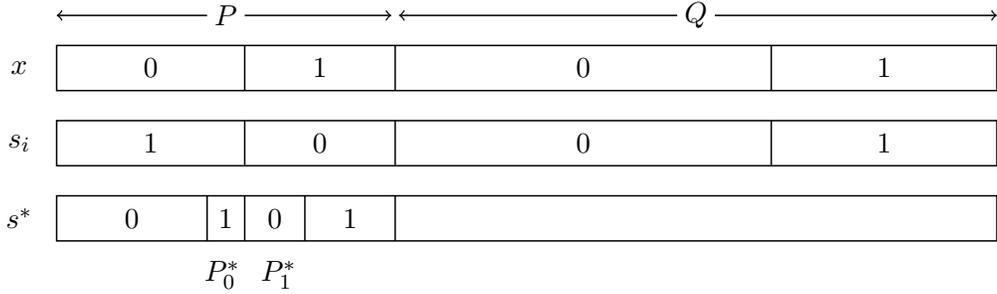


Figure 3: Strings $x$, $s_i$ and $s^*$ after permuting the positions.

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

$$n_1(x|_Q) = n_1(s_i|_Q). \tag{2}$$

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

$$n_1(s^*|_Q) - n_1(x|_Q) = n_1(x|_P) - n_1(s^*|_P). \tag{3}$$

Finally note that

$$n_1(s^*|_P) = |P_0^*| + n_1(x|_P) - |P_1^*|. \tag{4}$$

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

$$\mathcal{H}(s_i|_P, s^*|_P) = |P| - (|P_0^*| + |P_1^*|) = \mathcal{H}(x, s_i) - (|P_0^*| + |P_1^*|) \geq (1 + \epsilon)d - (|P_0^*| + |P_1^*|). \tag{5}$$

On the other hand, it holds that

$$\mathcal{H}(s_i|_Q, s^*|_Q) \geq |n_1(s^*|_Q) - n_1(s_i|_Q)| \overset{(2)}{=} |n_1(s^*|_Q) - n_1(x|_Q)|$$
$$\overset{(3)}{=} |n_1(x|_P) - n_1(s^*|_P)| \overset{(4)}{=} ||P_1^*| - |P_0^*||. \tag{6}$$

It follows that

$$d \geq \mathcal{H}(s_i, s^*) = \mathcal{H}(s_i|_P, s^*|_P) + \mathcal{H}(s_i|_Q, s^*|_Q)$$
$$\overset{(5),(6)}{\geq} (1 + \epsilon)d - (|P_0^*| + |P_1^*|) + ||P_1^*| - |P_0^*|| = (1 + \epsilon)d - 2\min(|P_0^*|, |P_1^*|).$$

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

**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 $j \in \{0, \dots, d\}$. Then, it holds that $\Pr[\mathcal{H}(x_j, s^*) \leq 2d - 2j] \geq \left(\frac{\epsilon}{3}\right)^{2j}$.*

*Proof.* We use induction on $j$. For $j = 0$ the claim follows from (1). Consider $j > 0$. By the induction hypothesis, we get

$$\Pr[\mathcal{H}(x_{j-1}, s^*) \leq 2d - 2j + 2] \geq \left(\frac{\epsilon}{3}\right)^{2j-2}. \tag{7}$$

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 $i \in [n]$. Since $\mathcal{H}(s^*, s_i) \leq d$, by the triangle inequality we get the following

$$|P_{j,0}| + |P_{j,1}| = \mathcal{H}(x_{j-1}, s_i) \leq \mathcal{H}(x_{j-1}, s^*) + \mathcal{H}(s^*, s_i) \leq 3d - 2j + 2 \leq 3d. \tag{8}$$

Then, we have

$$\Pr[\mathcal{H}(x_j, s^*) \leq 2d - 2j \mid \mathcal{H}(x_{j-1}, s^*) \leq 2d - 2j + 2] \geq \frac{|P_0^*| \cdot |P_1^*|}{|P_{j,0}| \cdot |P_{j,1}|} \geq \frac{\left(\frac{\epsilon d}{2}\right)^2}{\left(\frac{3d}{2}\right)^2} = \left(\frac{\epsilon}{3}\right)^2. \tag{9}$$

The first inequality holds through counting proper swaps among all possible swaps. The second inequality follows from Lemma 4.2 and (8). The claim follows by combining (7) and (9). $\qquad\square$

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.

Table 1 presents (rounded) values of $\epsilon$ for which the worst case bounds (with constants omitted) for the running times of algorithm from Theorem 4.1 and the algorithm of Misra et al. (2015) are equal, i.e., when $(3/\epsilon)^{2d} \cdot \log_2(1/p) = d^{2d}$ which gives $\epsilon = (3/d) \cdot (\log_2(1/p))^{\frac{1}{2d}}$. For $\epsilon$ greater than the values in Table 1 our algorithm can be faster than the previous one for instances with no solution at distance at most $d$ from $S$. Note that the effect of $p$ on the border value of $\epsilon$ is not very significant. However, a meaningful comparison of practical aspects of these two algorithms requires performing a series of experiments with actual implementations.

Table 1: Rounded values of $\epsilon = \frac{3}{d} \cdot \left( \log \frac{1}{p} \right)^{\frac{1}{2d}}$.

| $d$ $p$ | 10 | 15 | 20 | 25 |
|---|---|---|---|---|
| 0.5 | 0.3 | 0.2 | 0.15 | 0.12 |
| $10^{-10}$ | 0.357 | 0.225 | 0.164 | 0.129 |
| $10^{-20}$ | 0.370 | 0.230 | 0.167 | 0.131 |

## 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 $n^{\mathcal{O}(1/\epsilon^2 \cdot \log(1/\epsilon))} \cdot \text{poly}(m)$. 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 $(\{s_i\}_{i \in [n]}, k)$ of Minimax Approval Voting, i.e., $\text{OPT} = \max_{i \in [n]} \mathcal{H}(s, s_i)$,

**Theorem 5.1.** *There exists a randomized polynomial time algorithm which, for arbitrarily small fixed $p > 0$, given an instance $(\{s_i\}_{i \in [n]}, k)$ of* Minimax Approval Voting *and any $\epsilon > 0$ such that $\text{OPT} \geq \frac{122 \ln n}{\epsilon^2}$, reports a solution, which with probability at least $1 - p$ is at distance at most $(1 + \epsilon) \cdot \text{OPT}$ 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 use the 2-approximation of Caragiannis et al. (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 (10–13):

$$\text{minimize} \quad d \tag{10}$$

$$\sum_{j \in [m]} x_j = k \tag{11}$$

$$\sum_{\substack{j \in [m] \\ s_i[j]=1}} (1 - x_j) + \sum_{\substack{j \in [m] \\ s_i[j]=0}} x_j \leq d \quad \forall i \in [n] \tag{12}$$

$$x_j \in [0, 1] \quad \forall j \in [m] \tag{13}$$

The linear program (10–13) is a relaxation of the natural integer program for Minimax Approval Voting, obtained by replacing (13) by the discrete constraint $x_j \in \{0, 1\}$. Indeed, observe that $x_j$ corresponds to the $j$-th letter of the solution $x = x_1 \cdots x_m$, (11) states that $n_1(x) = k$, and (12) 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 $j \in [m]$ independently, we set $x[j] = 1$ with probability $x_j^*$. Note that $x$ does not need to contain $k$ ones. Let $y$ be any $k$-completion of $x$. The algorithm returns $y$.

---

**Pseudocode 2:** The algorithm from Theorem 5.1

---

1 Solve the LP (10–13) obtaining an optimal solution $(x_1^*, \ldots, x_m^*, d^*)$;
2 **for** $j \in \{1, 2, \ldots, m\}$ **do**
3     Set $x[j] \leftarrow 1$ with probability $x_j^*$ and $x[j] \leftarrow 0$ with probability $1 - x_j^*$

4 $y \leftarrow$ any $k$-completion of $x$;
5 **return** $y$

---

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 5.3 and 5.4, which can be shown using standard Chernoff bounds (see, e.g., Motwani & Raghavan, 1995, ch. 4.1).

**Theorem 5.2.** *(Motwani & Raghavan, 1995, ch. 4.1) Let $X_1, X_2, \ldots, X_n$ be $n$ independent random 0-1 variables such that for every $i \in [n]$ we have $\Pr[X_i = 1] = p_i$, for $p_i \in [0, 1]$. Let $X = \sum_{i=1}^n X_i$. Then,*

- *for any $0 < \epsilon \leq 1$ we have:*

$$\Pr[X > (1 + \epsilon) \cdot \mathbb{E}[X]] \leq \exp\left(-\tfrac{1}{3}\epsilon^2 \cdot \mathbb{E}[X]\right) \tag{14}$$

$$\Pr[X < (1 - \epsilon) \cdot \mathbb{E}[X]] \leq \exp\left(-\tfrac{1}{2}\epsilon^2 \cdot \mathbb{E}[X]\right) \tag{15}$$

- *for any $1 < \epsilon$ we have:*

$$\Pr[X > (1 + \epsilon) \cdot \mathbb{E}[X]] \leq \exp\left(-\tfrac{1}{3}\epsilon \cdot \mathbb{E}[X]\right) \tag{16}$$

$$\Pr[X < (1 - \epsilon) \cdot \mathbb{E}[X]] = 0 \tag{17}$$

**Lemma 5.3.** *It holds that* $\Pr\left[\mathcal{H}(x, S) > (1 + \tfrac{\epsilon}{2}) \cdot \mathrm{OPT}\right] \leq \tfrac{1}{4}$.

*Proof.* For every $i \in [n]$ we define a random variable $D_i$ that measures the distance between $x^*$ and $s_i$ by

$$D_i = \sum_{\substack{j \in [m] \\ s_i[j]=1}} (1 - x[j]) + \sum_{\substack{j \in [m] \\ s_i[j]=0}} x[j].$$

Note that $x[i]$ are independent 0-1 random variables. Using linearity of the expectation we obtain

$$
\begin{aligned}
\mathbb{E}[D_i] &= \mathbb{E}\left[\sum_{j \in [m], s_i[j]=1} (1 - x[j]) + \sum_{j \in [m], s_i[j]=0} x[j]\right] \\
&= \sum_{j \in [m], s_i[j]=1} (1 - \mathbb{E}[x[j]]) + \sum_{j \in [m], s_i[j]=0} \mathbb{E}[x[j]] \\
&= \sum_{j \in [m], s_i[j]=1} (1 - x_j^*) + \sum_{j \in [m], s_i[j]=0} x_j^* \quad \leq d^* \leq \mathrm{OPT}. \tag{18}
\end{aligned}
$$

Note that $D_i$ is a sum of $m$ independent 0-1 random variables $X_j = 1 - x[j]$ when $s_i[j] = 1$ and $X_j = x[j]$ otherwise. Denote $\delta = \epsilon \cdot \frac{\text{OPT}}{2\mathbb{E}[D_i]}$. We apply Chernoff bounds. For $\delta < 1$ we have

$$\Pr[D_i > \left(1 + \tfrac{\epsilon}{2}\right) \cdot \text{OPT}] \overset{(18)}{\leq} \Pr\left[D_i > \mathbb{E}[D_i] + \tfrac{\epsilon}{2} \cdot \text{OPT}\right] = \Pr\left[D_i > (1 + \delta) \cdot \mathbb{E}[D_i]\right]$$

$$\overset{(14)}{\leq} \exp\left(-\frac{1}{3}\left(\epsilon \cdot \frac{\text{OPT}}{2\mathbb{E}[D_i]}\right)^2 \mathbb{E}[D_i]\right) \overset{(18)}{\leq} \exp\left(-\frac{\epsilon^2 \cdot \text{OPT}}{12}\right).$$

In case $\delta \geq 1$ we proceed analogously, using the Chernoff bound (16) we get

$$\Pr[D_i > \left(1 + \tfrac{\epsilon}{2}\right) \cdot \text{OPT}] \overset{(16)}{\leq} \exp\left(-\frac{\epsilon \cdot \text{OPT}}{6}\right) \overset{1 > \epsilon}{\leq} \exp\left(-\frac{\epsilon^2 \cdot \text{OPT}}{12}\right).$$

Next, we use the union bound to get the claim

$$\Pr\left[\mathcal{H}(x, S) > (1 + \tfrac{\epsilon}{2}) \cdot \text{OPT}\right] = \Pr\left[\exists i \in [n] \quad D_i > \left(1 + \tfrac{\epsilon}{2}\right) \cdot \text{OPT}\right]$$

$$\leq n \cdot \exp\left(-\frac{\epsilon^2 \cdot \text{OPT}}{12}\right) \leq n \cdot \exp\left(-\frac{\frac{122 \ln n}{\text{OPT}} \cdot \text{OPT}}{12}\right) < n^{-9} \overset{n \geq 3}{<} \frac{1}{4}.$$

$\square$

**Lemma 5.4.** *It holds that* $\Pr\left[|n_1(x) - k| > \tfrac{\epsilon}{2} \cdot \text{OPT}\right] \leq \tfrac{1}{4}$.

*Proof.* First we note that

$$\mathbb{E}[n_1(x)] = \mathbb{E}\Big[\sum_{j \in [m]} x[j]\Big] = \sum_{j \in [m]} \mathbb{E}[x[j]] = \sum_{j \in [m]} x_j^* = k. \tag{19}$$

Pick an $i \in [n]$. Define the random variables

$$E_i = \sum_{j \in [m], s_i[j] = 1} (1 - x[j]), \quad F_i = \sum_{j \in [m], s_i[j] = 0} x[j].$$

Let $D_i = E_i + F_i$, as in the proof of Lemma 5.3. By (18) we have

$$\mathbb{E}[E_i] \leq \mathbb{E}[E_i] + \mathbb{E}[F_i] = \mathbb{E}[D_i] \leq \text{OPT} \tag{20}$$

$$\mathbb{E}[F_i] \leq \mathbb{E}[E_i] + \mathbb{E}[F_i] = \mathbb{E}[D_i] \leq \text{OPT} \tag{21}$$

Both $E_i$ and $F_i$ are sums of independent 0-1 random variables and we apply Chernoff bounds as follows. When $\frac{1}{4}\epsilon \cdot \frac{\text{OPT}}{\mathbb{E}[E_i]} \leq 1$ then using (14) and (15) we obtain

$$\Pr\left[\left|E_i - \mathbb{E}[E_i]\right| > \frac{1}{4}\epsilon \cdot \text{OPT}\right]$$

$$\overset{(14),(15)}{\leq} \exp\left(-\frac{1}{3} \cdot \frac{1}{16}\epsilon^2 \cdot \frac{(\text{OPT})^2}{\mathbb{E}^2[E_i]} \cdot \mathbb{E}[E_i]\right) + \exp\left(-\frac{1}{2} \cdot \frac{1}{16}\epsilon^2 \cdot \frac{(\text{OPT})^2}{\mathbb{E}^2[E_i]} \cdot \mathbb{E}[E_i]\right)$$

$$\overset{(20)}{\leq} 2 \cdot \exp\left(-\frac{1}{48}\epsilon^2 \cdot \text{OPT}\right),$$

otherwise $\left(\frac{1}{4}\epsilon \cdot \frac{\mathrm{OPT}}{\mathbb{E}[E_i]} > 1\right)$, using (16) and (17), we have

$$
\Pr\left[\left|E_i - \mathbb{E}[E_i]\right| > \frac{1}{4}\epsilon \cdot \mathrm{OPT}\right] \overset{(16),(17)}{\leq} \exp\left(-\frac{1}{3} \cdot \frac{1}{4}\epsilon \cdot \frac{\mathrm{OPT}}{\mathbb{E}[E_i]} \cdot \mathbb{E}[E_i]\right) + 0
$$

$$
\leq \exp\left(-\frac{1}{12}\epsilon \cdot \mathrm{OPT}\right) \overset{1 > \epsilon}{\leq} 2 \cdot \exp\left(-\frac{1}{48}\epsilon^2 \cdot \mathrm{OPT}\right).
$$

To sum up, in both cases we have shown that

$$
\Pr\left[\left|E_i - \mathbb{E}[E_i]\right| > \frac{\epsilon}{4} \cdot \mathrm{OPT}\right] \leq 2 \cdot \exp\left(-\frac{1}{48}\epsilon^2 \cdot \mathrm{OPT}\right). \tag{22}
$$

Similarly we show

$$
\Pr\left[\left|F_i - \mathbb{E}[F_i]\right| > \frac{\epsilon}{4} \cdot \mathrm{OPT}\right] \overset{(14),(15),(16),(17),(21)}{\leq} 2 \cdot \exp\left(-\frac{1}{48}\epsilon^2 \cdot \mathrm{OPT}\right). \tag{23}
$$

We see that

$$
n_1(x) = \sum_{j \in [m]} x[j] = n_1(s_i) - \sum_{j \in [m], s_i[j]=1}(1 - x[j]) + \sum_{j \in [m], s_i[j]=0} x[j] = n_1(s_i) - E_i + F_i \tag{24}
$$

and hence it holds

$$
\mathbb{E}[n_1(x)] = n_1(s_i) - \mathbb{E}[E_i] + \mathbb{E}[F_i]. \tag{25}
$$

Additionally we will use

$$
\forall x, y \in \mathbb{R} \quad |x - y| > a \implies |x| > a/2 \ \lor \ |y| > a/2. \tag{26}
$$

Now we can write

$$
\Pr\left[\left|n_1(x) - k\right| > \tfrac{1}{2}\epsilon \cdot \mathrm{OPT}\right] \overset{(19)}{=} \Pr\left[\left|n_1(x) - \mathbb{E}[n_1(x)]\right| > \tfrac{1}{2}\epsilon \cdot \mathrm{OPT}\right]
$$

$$
\overset{(24),(25)}{=} \Pr\left[\left|n_1(s_i) - E_i + F_i - n_1(s_i) + \mathbb{E}[E_i] - \mathbb{E}[F_i]\right| > \tfrac{1}{2}\epsilon \cdot \mathrm{OPT}\right]
$$

$$
\overset{(26)}{\leq} \Pr\left[\left|E_i - \mathbb{E}[E_i]\right| > \tfrac{1}{4}\epsilon \cdot \mathrm{OPT} \ \lor \ \left|F_i - \mathbb{E}[F_i]\right| > \tfrac{1}{4}\epsilon \cdot \mathrm{OPT}\right]
$$

$$
\leq \Pr\left[\left|E_i - \mathbb{E}[E_i]\right| > \tfrac{1}{4}\epsilon \cdot \mathrm{OPT}\right] + \Pr\left[\left|F_i - \mathbb{E}[F_i]\right| > \tfrac{1}{4}\epsilon \cdot \mathrm{OPT}\right]
$$

$$
\overset{(22),(23)}{\leq} 4 \cdot \exp\left(-\tfrac{1}{48}\epsilon^2 \cdot \mathrm{OPT}\right) \overset{\mathrm{assum.}}{\leq} 4 \cdot \exp\left(-\tfrac{122}{48}\ln n\right) \overset{n \geq 3}{<} \tfrac{1}{4}.
$$

$\square$

We can finish the proof of Theorem 5.1. By Lemmas 5.3 and 5.4 with probability at least $1/2$ both $\mathcal{H}(x, S) \leq (1 + \frac{1}{2}\epsilon) \cdot \mathrm{OPT}$ and $\mathcal{H}(y, x) = |n_1(x) - k| \leq \frac{1}{2}\epsilon \cdot \mathrm{OPT}$. By the triangle inequality this implies that $\mathcal{H}(y, S) \leq (1 + \epsilon) \cdot \mathrm{OPT}$, with probability at least $1/2$ as required.

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

**Theorem 5.5.** *For each $\epsilon > 0$ we can find $(1 + \epsilon)$ approximation solution for the* MINIMAX APPROVAL VOTING *problem in time $n^{\mathcal{O}\left(\frac{\log 1/\epsilon}{\epsilon^2}\right)} \cdot \mathrm{poly}(m)$ with probability at least $1 - r$, for any fixed $r > 0$.*

*Proof.* First we run the 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 OPT $\geq d$, even the initial solution is at distance at most $(1 + \epsilon)d \leq (1 + \epsilon) \cdot$ OPT from $S$. Otherwise, at some point $d' =$ OPT and we get a $(1 + \epsilon)$-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 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

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

$\square$

Let us comment on the practicality of the PTAS presented in this section. Although the asymptotic worst-case complexity is better than in the case of the previous PTAS (Byrka & Sornat, 2014), the large constants hidden in the exponents of the function describing the running time still make it far from being practical. A further algorithm engineering research effort can help to turn our ideas into a useful implementation.

## 6. Concluding Remarks and Open Questions

There are examples of theoretical work which was next turned into practical software by means of non-trivial algorithm engineering effort. See, e.g., the algorithm of Tamaki (2017) based on work of Bouchitte and Todinca (2001), which solved all 100 instances of exact treewidth challenge at the PACE 2017 competition (Dell, Komusiewicz, Talmon, & Weller, 2017). Similarly, we believe that our techniques, possibly augmented with additional ideas, may be used in an efficient implementation. However, our lower bound shows an obstacle which any such implementation has to face.

Using approximation and randomized algorithms for finding committees in multi-winner elections appears to be controversial. The outcome can be non-optimal and additionally it can be different for different random bits. One can see an approximation algorithm as a new voting rule. In fact, SEQUENTIAL PROPORTIONAL APPROVAL VOTING (Thiele, 1895), that was used briefly in Sweden during the early 1900s (Aziz et al., 2017b), is a greedy algorithm that gives an approximate solution to PROPORTIONAL APPROVAL VOTING. For more discussion and examples of approximation algorithms being full-fledged voting rule we delegate to the following papers (Caragiannis et al., 2012; Caragiannis, Kaklamanis, Karanikolas, & Procaccia, 2014; Skowron, Faliszewski, & Slinko, 2015; Aziz et al., 2017b;

Elkind et al., 2017). While using approximation or randomization in domains similar political elections may appear controversial, multi-winner voting has much more diverse applications. Examples include cost-minimization or gain-maximization, see e.g., a recent paper by Byrka, Skowron, and Sornat (2018) for connections between multi-winner elections and facility location problems. Finally, randomized and approximation algorithms seem to be well justified for high-frequency decisions (Aziz, Bogomolnaia, & Moulin, 2017a), e.g., online scheduling or online systems.

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, Lokshtanov, Pilipczuk, Pilipczuk, and Saurabh (2016) 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 $k \in \{0, 1, \ldots, m\}$. 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

## References

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

Aziz, H., Bogomolnaia, A., & Moulin, H. (2017a). Fair Mixing: The Case of Dichotomous Preferences. *CoRR, abs/1712.02542*.

Aziz, H., Brill, M., Conitzer, V., Elkind, E., Freeman, R., & Walsh, T. (2017b). Justified Representation in Approval-based Committee Voting. *Social Choice and Welfare*, *48*(2), 461–485.

Baumeister, D., Bohnlein, T., Rey, L., Schaudt, O., & Selker, A. (2016). Minisum and Minimax Committee Election Rules for General Preference Types. In *Proceedings of 22nd European Conference on Artificial Intelligence, ECAI 2016*, Vol. 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 1656–1657. IOS Press.

Betzler, N., Slinko, A., & Uhlmann, J. (2013). On the Computation of Fully Proportional Representation. *Journal of Artificial Intelligence Research*, *47*, 475–519.

Bouchitte, V., & Todinca, I. (2001). Treewidth and Minimum Fill-in: Grouping the Minimal Separators. *SIAM Journal of Computing*, *31*(1), 212–232.

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

Brams, S. J., Kilgour, D. M., & Sanver, M. R. (2007b). A Minimax Procedure for Negotiating Multilateral Treaties. In Avenhaus, R., & Zartman, I. W. (Eds.), *Diplomacy Games*, pp. 265–282. Springer Berlin, Heidelberg.

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

Byrka, J., Skowron, P., & Sornat, K. (2018). Proportional Approval Voting, Harmonic k-median, and Negative Association. In *Proceedings of 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, pp. 26:1–26:14.

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

Caragiannis, I., Covey, J. A., Feldman, M., Homan, C. M., Kaklamanis, C., Karanikolas, N., Procaccia, A. D., & Rosenschein, J. S. (2012). On the Approximability of Dodgson and Young Elections. *Artificial Intelligence*, *187*, 31–51.

Caragiannis, I., Kaklamanis, C., Karanikolas, N., & Procaccia, A. D. (2014). Socially Desirable Approximations for Dodgson's Voting Rule. *ACM Transactions on Algorithms*, *10*(2), 6:1–6:28.

Caragiannis, I., Kalaitzis, D., & Markakis, E. (2010). Approximation Algorithms and Mechanism Design for Minimax Approval Voting. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010*, pp. 737–742.

Chamberlin, J. R., & 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. *Communications of the ACM*, *53*(3), 84–94.

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

Cygan, M., Kowalik, Ł., Socała, A., & Sornat, K. (2017). Approximation and Parameterized Complexity of Minimax Approval Voting. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 459–465.

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

Dell, H., Komusiewicz, C., Talmon, N., & Weller, M. (2017). The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration. In *Proceedings of 12th International Symposium on Parameterized and Exact Computation, IPEC 2017*, pp. 30:1–30:12.

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

Elkind, E., Faliszewski, P., Skowron, P., & Slinko, A. (2017). Properties of Multiwinner Voting Rules. *Social Choice and Welfare*, *48*(3), 599–632.

Faliszewski, P., Skowron, P., Slinko, A., & Talmon, N. (2017). Multiwinner Voting: A New Challenge for Social Choice Theory. In Endriss, U. (Ed.), *Trends in Computational Social Choice*, chap. 2, pp. 27–47. AI Access.

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

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

Impagliazzo, R., & Paturi, R. (2001). On the Complexity of k-SAT. *Journal of Computer and System Sciences*, *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., & Sanver, R. M. (Eds.), *Handbook on Approval Voting*, pp. 105–124. Springer Berlin, Heidelberg.

Laslier, J.-F., & Sanver, R. M. (2010). *Handbook on Approval Voting*. Studies in Choice and Welfare. Springer Berlin, Heidelberg.

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

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

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

Liu, H., & 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*, pp. 341–349.

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

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

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

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

Misra, N. (2016) personal communication.

Misra, N., Nabeel, A., & 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*, pp. 97–105.

Motwani, R., & Raghavan, P. (1995). *Randomized Algorithms.* Cambridge University Press.

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.

Rawls, J. (1971). *A Theory of Justice.* Harvard Univeristy Press.

Sanchez-Fernandez, L., & Fisteus, J. A. (2017). Monotonicity Axioms in Approval-based Multi-winner Voting Rules. *CoRR, abs/1710.04246.*

Skowron, P., Faliszewski, P., & Lang, J. (2016). Finding a Collective Set of Items: From Proportional Multirepresentation to Group Recommendation. *Artificial Intelligence, 241*, 191–216.

Skowron, P., Faliszewski, P., & Slinko, A. M. (2015). Achieving Fully Proportional Representation: Approximability Results. *Artificial Intelligence, 222*, 67–103.

Tamaki, H. (2017). Positive-Instance Driven Dynamic Programming for Treewidth. In *Proceedings of 25th Annual European Symposium on Algorithms, ESA 2017*, pp. 68:1–68:13.

Thiele, T. N. (1895). Om Flerfoldsvalg. In *Oversigt over det Kongelige Danske Videnskabernes Selskabs Forhandlinger*, pp. 415–441.