Improving the Security of United States Elections with Robust Optimization

Braden L. Crimmins[1], J. Alex Halderman[1], Bradley Sturt[2]

[1]Computer Science and Engineering, University of Michigan, Ann Arbor
[2]Information and Decision Sciences, University of Illinois Chicago

Abstract

For more than a century, election officials across the United States have inspected voting machines before elections using a procedure called Logic and Accuracy Testing (LAT). This procedure consists of election officials casting a test deck of ballots into each voting machine and confirming the machine produces the expected vote total for each candidate. We bring a scientific perspective to LAT by introducing the first formal approach to designing test decks with rigorous security guarantees. Specifically, our approach employs robust optimization to find test decks that are guaranteed to detect any voting machine misconfiguration that would cause votes to be swapped across candidates. Out of all the test decks with this security guarantee, our robust optimization problem yields the test deck with the minimum number of ballots, thereby minimizing implementation costs for election officials. To facilitate deployment at scale, we develop a practically efficient exact algorithm for solving our robust optimization problems based on the cutting plane method. In partnership with the Michigan Bureau of Elections, we retrospectively applied our approach to all 6928 ballot styles from Michigan's November 2022 general election; this retrospective study reveals that the test decks with rigorous security guarantees obtained by our approach require, on average, only 1.2% more ballots than current practice. Our approach has since been piloted in real-world elections by the Michigan Bureau of Elections as a low-cost way to improve election security and increase public trust in democratic institutions.

First version: August 4, 2023. Revisions submitted on May 21, 2024 and September 3, 2024. Accepted for publication on September 27, 2024.

1 Introduction

Computerized voting machines are widely used to scan ballots and determine election outcomes throughout the United States and around the world. Voting machines are used instead of hand counting because voters are often invited to participate in a large number of contests in an election—including political offices from the President to local school boards—which causes hand counting to be impractically costly and time consuming. In this paper, we develop a low-cost approach to reducing the security risks of voting machines and improving public trust in democratic institutions by drawing on techniques from the field of robust optimization.

1.1 Background

For voting machines that scan ballots to count votes accurately, they must be configured with correct mappings between the voting targets on the ballot—i.e., the boxes or ovals that voters mark—and the candidates who should receive the votes. If a voting machine is configured with an incorrect mapping, then the machine may count votes for the wrong candidates. As illustrated by the following examples, voting machines misconfigurations can produce dramatically wrong vote totals and damage public trust in elections:

- During the 2020 election, voting machines in Antrim County, Michigan were accidentally misconfigured with mappings that caused votes for Republicans to be tallied for Democrats and votes for Democrats to go uncounted [14]. The erroneous vote totals announced as a result of this flaw received widespread media coverage [8, 1], and this incident served as the basis for a draft executive order, later obtained by the Congressional committee investigating the events of January 6, 2021, that would have directed the Secretary of Defense to seize voting machines [27].

- Similar accidental misconfigurations affected announced election results in Pennsylvania [10] and Georgia [12] in the past five years. Although the errors were quickly caught and corrected, they similarly resulted in the initial publication of incorrect vote totals and generated significant negative publicity for the affected jurisdictions. In a particularly recent example, an accidental misconfiguration in Northampton County, Pennsylvania during their November 2023 election caused votes to be swapped across two judge contests, leading to voter confusion and long lines on election day [25, 28].

- Misconfigurations could also be induced deliberately by adversaries with very little technical expertise. Indeed, a group that contends the outcome of the 2020 presidential election was fraudulent recently released a video that demonstrates exactly how one could strategically induce these misconfigurations to manipulate future election outcomes [11]. Such deliberate manipulations would allow an adversary to sow doubt in election systems, influence who wins prominent political offices, and—in at least 24 states [22]—directly affect the passage of laws on matters ranging from environmental policy to abortion rights.

Past work has sought to address the potential dangers of compromised voting machines through post-election interventions such as risk-limiting audits and cryptographic systems that make announced results publicly verifiable [19, 6]. These post-election procedures are increasingly being implemented in the United States and have received attention in popular media [23]. However, no prior work has developed a procedure that is guaranteed to detect important classes of possible attacks *before* an election takes place. Such a procedure could help safeguard election integrity and public confidence by detecting attacks before they affect reported vote totals. Our paper develops a

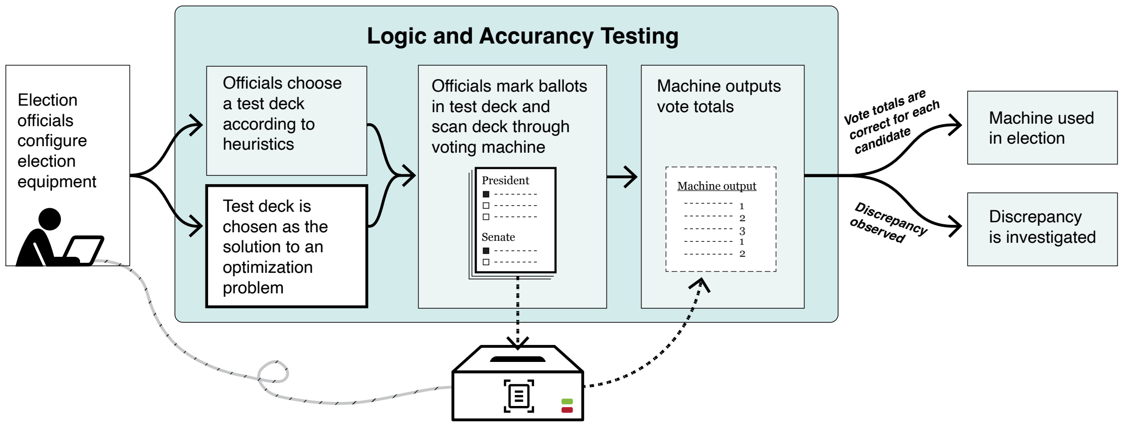

Logic and Accurancy Testing
Election officials configure election equipment
Officials choose a test deck according to heuristics
Test deck is chosen as the solution to an optimization problem
Officials mark ballots in test deck and scan deck through voting machine
President
Senate
Machine outputs vote totals
Machine output
Vote totals are correct for each candidate
Discrepancy observed
Machine used in election
Discrepancy is investigated

Figure 1: Visualization of Logic and Accuracy Testing (LAT). The procedure is conducted chronologically from left to right on voting machines before each election. The modification to LAT proposed in this paper is denoted by the white box with the text "Test deck is chosen as the solution to an optimization problem".

rigorous and low-cost pre-election defense against misconfiguration-based cyberattacks by applying robust optimization to a widely-used testing procedure called Logic and Accuracy Testing (LAT).

1.2 Logic and Accuracy Testing

For more than a century, election officials throughout the United States have used LAT to inspect voting machines prior to elections. The idea behind LAT is simple: officials prepare a set of ballots with known votes—dubbed a test deck—then cast the ballots through each voting machine and confirm that the machine outputs the expected tallies for all candidates (see Figure 1). Any discrepancy indicates a potential malfunction, which can be addressed before the machine is used to count real votes. LAT was initially developed in the early 1900s to protect against breakdowns of mechanical lever-based voting machines and is today required by law before each election in all fifty states [15, 29].

Despite the widespread use of LAT, no prior work has used LAT for detecting attacks on modern computerized voting machines. In fact, LAT is not an obvious candidate for securing modern elections; it cannot, for example, detect malicious alterations to a voting machine's software that cause the voting machine to operate fraudulently only after testing has concluded.[1] Nonetheless, LAT has a number of properties that make it potentially attractive for election security. First, the legally mandated use of LAT across the United States means that repurposing this procedure as a modern security tool would require little investment on already-overburdened election administrators. Second, the fact that LAT is performed prior to elections means that it is well-situated to detect cyberattacks before they affect the public. Third, developing sophisticated cyberattacks that cannot be detected by LAT requires technical capabilities that are out of reach for many would-be adversaries. In particular, we show in this paper that LAT has the potential to be an effective defense against less sophisticated (yet still practically significant) classes of attacks that are based on deliberate misconfiguration of voting machines.

The set of misconfigurations which would be detected by LAT hinges on the design of the test deck, i.e., the decision of which voting targets to fill out on each ballot. Until now, test decks

[1]Such manipulations are sometimes called "Volkswagen attacks", in reference to the 2015 Volkswagen emissions scandal wherein vehicle motors were programmed to reduce their emission levels only when the vehicles were undergoing testing for compliance with environmental efficiency regulations [9].

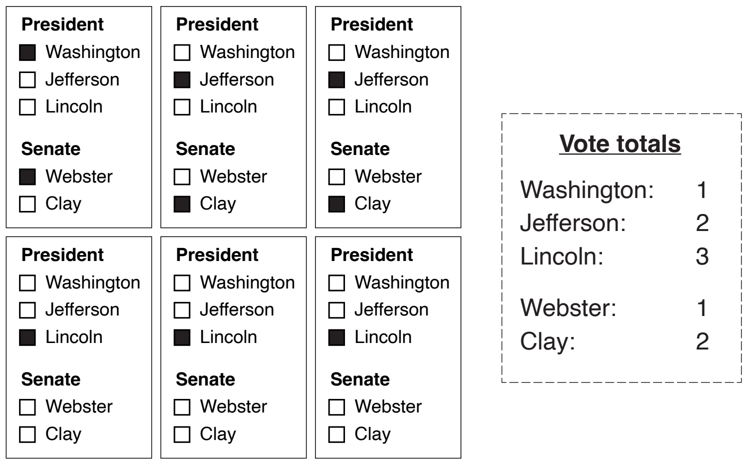


Figure 2: A test deck composed of six ballots for a simple election with two contests. The first contest is a presidential contest with three candidates; the second contest is a senatorial contest with two candidates. In each contest, a voter is allowed to vote for at most one candidate.

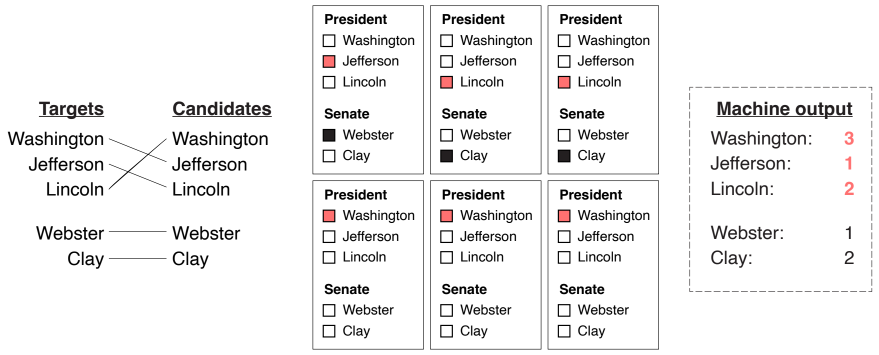
throughout the United States have been designed following simple heuristics that are based on human intuition [29]. For example, Figure 2 shows an example of a test deck constructed by a common heuristic that gives each candidate within each contest a different number of votes. However, Figure 8 shows the output of voting machines using the test deck from Figure 2 under three examples of misconfigured mappings between voting targets and candidates, including one misconfiguration which this test deck would not detect. This demonstrates that this simple heuristic is not guaranteed to secure the voting machine from misconfiguration attacks. If each candidate on the ballot received a different number of votes, then all misconfigured mappings between voting targets and candidates would be detected, but this strategy for designing test decks is not used in practice because it requires impractically many ballots for real-world elections (see Appendix A). The difficulty of marking and scanning test decks scales with the number of ballots included, so short test decks are imperative for practical implementation.

1.3 Contributions

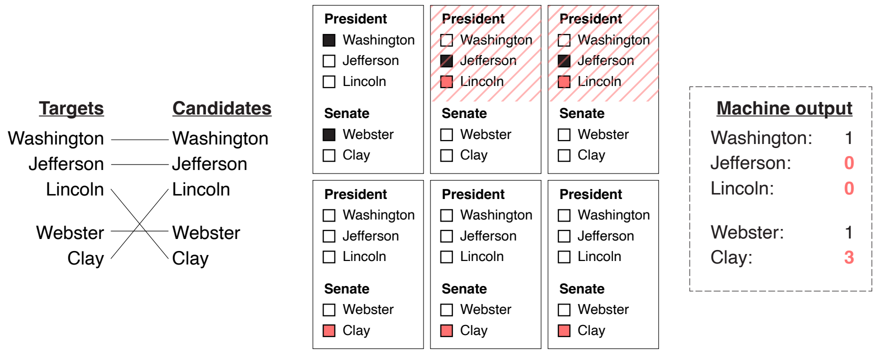
We bring a scientific perspective to LAT by introducing the first formal approach to designing test decks for LAT with rigorous security guarantees. Specifically, our approach employs mathematical optimization—rather than heuristics—to find test decks that are guaranteed to detect any misconfiguration that swaps votes between candidates. Moreover, out of all the test decks that are guaranteed to detect these swaps, our approach yields a test deck with the minimum number of ballots, thereby minimizing implementation difficulties for election officials.

In greater detail, our approach to designing test decks consists of constructing and solving a robust optimization problem [5, 7]. The input to the robust optimization problem is a "ballot style," which for our purposes means the set of contests that appear on a given ballot, the set of candidates who are running in each of those contests, the maximum number of candidates that a voter is allowed to select in each contest, and the correct mapping of voting targets to candidates.[2] The output of

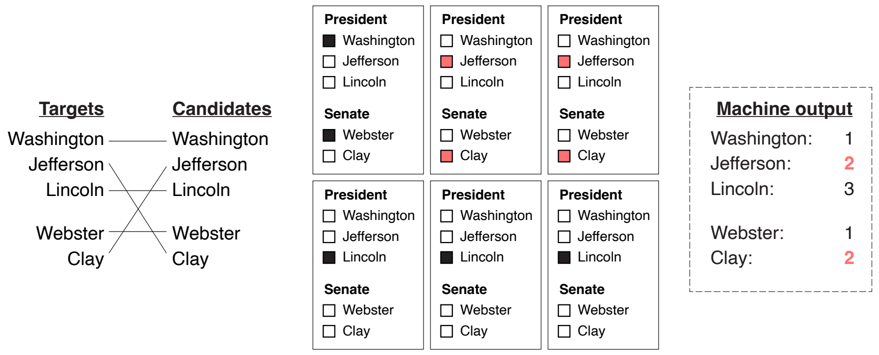
[2]Because the contests available on a ballot depend on granular political subdivisions like county, municipality, legislative district, and school district, there are often many thousands of ballot styles used across the different jurisdictions of a state in any given election. Voting machines across the state are configured separately for each of the different ballot styles, and a different test deck must be used to evaluate each such configuration.



(a)



(b)



(c)

Figure 3: Each example shows a misconfiguration of the mapping between voting targets and candidates (left), the misconfigured voting machine's interpretation of the test deck from Figure 2 (center), and the vote tally that is output by the misconfigured voting machine (right). The color red indicates the aspects of the interpretation of the test deck and the machine output that are impacted by the misconfiguration of the voting machine. Diagonal lines through a contest indicate that the filled-out ballot is interpreted as containing an overvote in that contest, in which case the voting machine interprets the filled-out ballot as if no candidates were selected in that contest. **(a)** *The misconfiguration is detected because the output of the voting machine includes incorrect vote totals for Washington, Jefferson, and Lincoln.* **(b)** *The misconfiguration is detected because the output of the voting machine includes incorrect vote totals for Jefferson, Lincoln, and Clay.* **(c)** *The misconfiguration is not detected because the output of the voting machine includes correct vote totals for all candidates (see Figure 2).*

the robust optimization problem is the design of a minimum-length test deck that is guaranteed to detect whether a voting machine has an incorrect bijective mapping from candidates to voting targets for the ballot style in question. The robust optimization problem is stated formally in §3.3.

One of the key difficulties in solving our robust optimization problem lies in the large number of incorrect bijective mappings from candidates to voting targets. In a ballot style with N candidates and voting targets, there are $N! - 1$ possible ways that voting targets can be swapped, one swap for each bijection over candidates less the single correct bijection. In United States elections, the number of candidates across the contests of a ballot style is often greater than one hundred. Consequently, formulating our robust optimization problem often requires more than $100! - 1 \approx 10^{157}$ constraints, a number far greater than the estimated number of particles in the observable universe [30]. An optimization problem that explicitly encodes all possible swaps thus cannot be represented nor solved on any extant computer for many real-world elections. It is currently unknown whether the robust optimization problem is NP-hard, and it is unknown whether there exists a mixed-integer linear programming reformulation of the robust optimization problem of size that is polynomial in the number of candidates N.

We contend with the above computational challenge by developing an exact algorithm for the robust optimization problem inspired by the cutting plane method (see §4.1). The cutting plane method is a classical technique for solving optimization problems with many constraints by solving a sequence of optimization problems with small numbers of constraints. In our setting, each iteration of the cutting plane method solves a relaxation of the robust optimization problem that contains a small subset of the $N! - 1$ swaps. If the optimal test deck of the relaxed optimization problem detects all of the $N! - 1$ swaps of the original problem, then the algorithm terminates. Otherwise, the algorithm finds a swap that is undetected by the optimal test deck of the relaxed problem, adds the undetected swap into the relaxed problem, and then solves the relaxed problem again. This process repeats until a feasible solution for the original robust optimization problem is obtained.

To make the cutting plane method terminate in practical computation times in real-world elections, we make a number of novel algorithmic developments. First, we reformulate the relaxed optimization problem as well as the problem of finding an undetected swap as mixed-integer linear optimization problems (see §4.2). These reformulations enable the cutting plane method to be easily implemented using widely available open-source and commercial optimization software such as Gurobi and Mosek. Second, we offer a variety of theoretically-justified improvements to our mixed-integer linear optimization formulations (see §5) that aim to decrease the number of iterations and decrease the per-iteration computation time of the cutting plane method. These improvements include dynamically identifying and removing unnecessary decision variables from the mixed-integer linear optimization problems (§5.1), adding constraints that impose the structure of optimal test decks into the mixed-integer linear optimization problems (§5.2 and §5.3), developing a combinatorial framework for identifying which swap to add to the relaxed optimization problem in each iteration (§5.4), and combining all contests that are not competitive (§5.5). In Appendix C, we demonstrate via experiments on synthetic elections that each of our improvements yields significant decreases in the computation time and number of iterations of the cutting plane method.

We conclude by showcasing the value of our robust optimization approach in application to real world elections. In partnership with the Michigan Bureau of Elections, we applied our approach to each of the state's 6928 ballot styles from the November 2022 general election. Our results for this election (see §6) reveal that our approach only required a 1.2% average increase in the number of test ballots compared to current practice across the state's 6928 ballot styles. Hence, our approach can be deployed with minimal financial cost or operational overhead while providing significant security benefits to election jurisdictions. Moreover, our cutting plane method for solving the robust optimization problems enabled our approach to obtain optimal test decks for all 6928 ballot styles in less than seven hours. These findings demonstrate that our cutting plane method can find optimal test decks for all of the ballot styles across a state in computation times that are practical from the perspective of election officials. Our approach described in this paper has been piloted by the Michigan Bureau of Elections in real-world elections during the summer of 2023, and we hope that our approach will be adopted by more states and countries in upcoming elections as a low-cost tool to improving the security and increasing public confidence in election outcomes.

An open source portion of the code from this paper is available at https://github.com/ballotiq/deck-checker.

2 Vulnerabilities of Existing Heuristics for Designing Test Decks

Our proposed approach to designing test decks with rigorous security guarantees is presented in §3. To motivate our approach, we begin in this section by describing three examples of misconfiguration attacks against United States voting machines. We show in each of the three examples how the attack could be strategically deployed by an adversary to undermine public trust or change the outcome of an election. Finally, we show how the examples of attacks could evade detection by LAT when test decks are designed by commonly used heuristics.

Swaps of Individual Candidates. Suppose that the goal of an adversary is to decrease the number of votes received by a specific candidate in a high-stakes contest near the top of the ballot (such as a presidential contest). In this case, an example of a misconfiguration that would be appealing to the adversary is one that swaps the voting target of the specific candidate with the target of a candidate from a contest that is lower on the ballot (such as the contest to elect a sanitation commissioner). Because fewer people vote in downballot contests [17], this misconfiguration could result in the adversary's disfavored presidential candidate receiving fewer votes than they should. Moreover, if the test deck for LAT is designed using a common heuristic in which a single ballot contains votes for the first candidate in each contest, two ballots contain votes for the second candidate in each contest, and so on, then LAT would not detect any misconfiguration that swaps the targets for two candidates at corresponding indices in their respective contests. An example of a test deck constructed by this common heuristic is shown in Figure 2, and the misconfiguration depicted in Figure 8c is an example of such a swap that goes undetected, since it swaps the second candidate in the presidential contest with the second candidate in the senatorial contest.

Swaps of Entire Contests. In many states, elections put certain yes-or-no questions—commonly called initiatives, proposals, or referendums—directly to voters. The effect of these contests range from modifying a state's constitution on matters such as abortion rights [26] and environmental policy [4, 18] to recalling sitting politicians from their office [20]. If an adversary wished to swap the outcome of two such contests, they could misconfigure the voting machine to swap the voting targets for 'yes' and for 'no' between the two contests. Moreover, if LAT is conducted with a test deck that includes the same number of votes for 'yes' and the same number of votes for 'no' in each of the two contests—which is the case under every common heuristic for test deck preparation used today [29]—then this misconfiguration would not be detected by LAT (see §5.3). This attack could thus be used to ensure a favored proposal passes or a disfavored proposal fails, and would allow an adversary to directly influence the laws or constitution of a jurisdiction.

Deliberately Flawed Test Decks. It is common for jurisdictions to contract outside vendors to configure their voting machines as well as design the test decks used to conduct LAT. If this vendor is untrustworthy, they could misconfigure the machine according to their own preference, then deliberately construct a test deck which would fail to detect the modification. Indeed, we show in Appendix B that a vendor has significant freedom in the misconfiguration they choose, even when the test deck they produce is constrained by some of the most stringent legal requirements in use by states today.

In the following section, we introduce an approach to designing test decks that enables LAT to become a rigorous pre-election defense against an important class of misconfiguration attacks. This class includes, among many others, the three examples of attacks described above.

3 Robust Logic and Accuracy Testing

In this section, we introduce Robust Logic and Accuracy Testing (RLAT), an optimization-based framework for designing test decks in LAT with rigorous security guarantees. This section has the following organization. §3.1 develops the terminology and mathematical notation that will be used throughout the paper. §3.2 presents a general formulation of RLAT and discusses its value from the perspective of various stakeholders in United States elections. §3.3 uses the RLAT framework to derive our robust optimization problem (RO-Σ) for finding a minimum-length test deck that will detect whether a voting machine is misconfigured to swap votes between candidates. §3.4 establishes the fundamental structural properties of test decks that are feasible for the robust optimization problem (RO-Σ).

3.1 Preliminaries

A ballot style is composed of a set of contests $\mathcal{C} \triangleq \{1,\dots,C\}$ and a set of candidates $\mathcal{N} \triangleq \{1,\dots,N\}$. For each contest $c \in \mathcal{C}$, we let $\mathcal{N}_c \subseteq \mathcal{N}$ denote the subset of candidates that appear in contest c, and we let v_c denote the maximum number of candidates in that contest that may be legally selected by a voter. For example, for a contest that corresponds to the senatorial election, the set $\mathcal{N}_c$ would contain the indices of the candidates that are running for Senator, and the equality $v_c = 1$ would denote that each voter is permitted to select at most one candidate in the contest. In a contest for a local school board with five vacancies, we would alternatively have the equality $v_c = 5$. We assume that each candidate $i \in \mathcal{N}$ appears in exactly one contest. We say that a contest c is noncompetitive if the maximum number of votes v_c is equal to the number of candidates $|\mathcal{N}_c|$ in the contest.[3] In real-world elections such as those from Michigan (see Figure 3), the number of contests in each ballot style typically satisfies $15 \le C \le 40$, and the number of candidates in each ballot style typically satisfies $60 \le N \le 120$.

A ballot refers to a physical document that contains a box or oval beside each candidate, termed targets, that are used by voters to record their choices. With a slight abuse of notation, we denote the targets on a ballot by $\mathcal{N} \triangleq \{1,\dots,N\}$, where each target $i \in \mathcal{N}$ refers to the box or oval that is beside candidate $i \in \mathcal{N}$. A filled-out ballot is represented by a subset of targets $\beta \subseteq \mathcal{N}$, with the interpretation that the filled-out ballot satisfies $i \in \beta$ if and only if the filled-out ballot selected target i. It follows that the number of targets beside candidates in contest $c \in \mathcal{C}$ that are selected by a filled-out ballot $\beta \subseteq \mathcal{N}$ is equal to $|\mathcal{N}_c \cap \beta|$. A deck refers to any finite-length sequence of filled-out ballots $(\beta_1,\dots,\beta_B)$.

[3]Noncompetitive contests often arise when an incumbent to some local office runs unopposed for re-election. This is especially common in states which elect judges, since there is a strong normative prohibition against challenging a sitting judge's re-election bid [24].

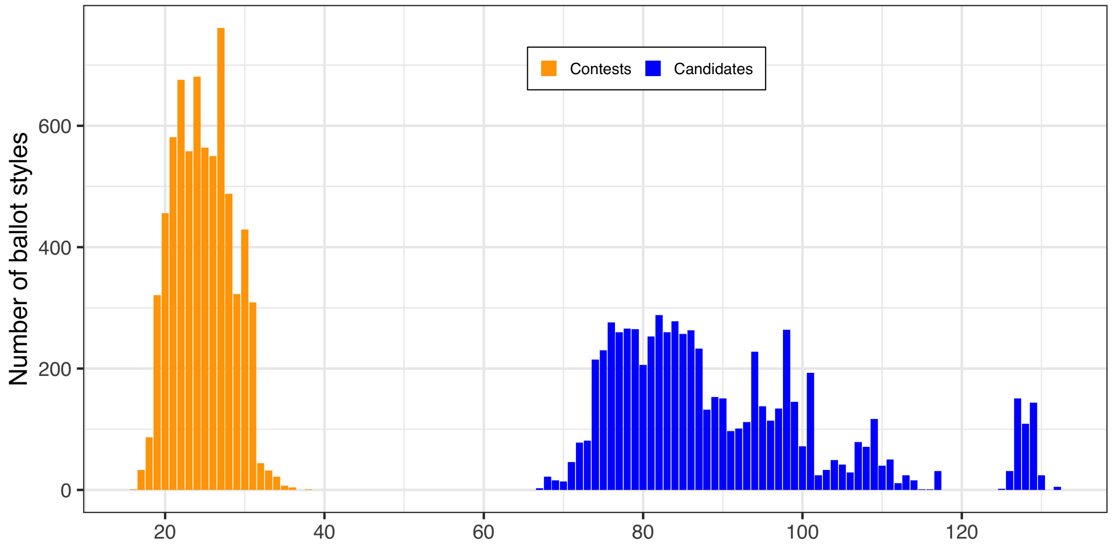


Figure 4: Histogram of the total number of contests (orange) and total number of candidates (blue) that appeared across the 6928 ballot styles in Michigan's November 2022 general election.

When a voting machine operates correctly, the machine will receive a deck of filled-out ballots as its input, and the machine will output the total number of targets that are selected for each candidate in the filled-out ballots that do not have an overvote in that candidate's contest. For any input deck $(\beta_1, \ldots, \beta_B)$, we denote the output of a voting machine that operates correctly by the vector-valued function

$$T^*(\beta_1, \ldots, \beta_B) \equiv (T_1^*(\beta_1, \ldots, \beta_B), \ldots, T_N^*(\beta_1, \ldots, \beta_B)),$$

with the output for each candidate $i \in \mathcal{N}_c$ in each contest $c \in \mathcal{C}$ defined as

$$T_i^*(\beta_1, \ldots, \beta_B) \triangleq \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\mathcal{N}_c \cap \beta_b| \leq v_c\right\}.$$

In the above definition, and throughout the rest of this paper, we let $\mathbb{I}\{\cdot\}$ represent the indicator function that is equal to one if $\cdot$ is true and is equal to zero if $\cdot$ is false. The inclusion $i \in \beta_b$ holds if and only if filled-out ballot β_b has selected the target that is beside candidate i, and the inequality $|\mathcal{N}_c \cap \beta_b| \leq v_c$ holds if and only if filled-out ballot β_b has selected at most v_c of the targets that are beside the candidates in contest c. In other words, the inequality $|\mathcal{N}_c \cap \beta_b| \leq v_c$ holds if and only if filled-out ballot β_b is interpreted by the voting machine that operates correctly as not containing an overvote in contest c. For notational convenience, we denote the set of ballots that do not overvote any contest by $\mathscr{B} \triangleq \{\beta \subseteq \mathcal{N} : |\mathcal{N}_c \cap \beta| \leq v_c \; \forall c \in \mathcal{C}\}$.

Remark 1. If the filled-out ballots in an input deck do not contain overvotes, then the output of the voting machine that operates correctly will equal the number of filled-out ballots that select the target associated with each candidate. In other words, if $\beta_1, \ldots, \beta_B \in \mathscr{B}$, then the equality $T_i^*(\beta_1, \ldots, \beta_B) = |\{b \in \{1, \ldots, B\} : i \in \beta_b\}|$ holds for each candidate $i \in \mathcal{N}$.

To represent the output of a specific voting machine that may or may not be operating correctly, we use the vector-valued function

$$\widehat{T}(\beta_1, \ldots, \beta_B) \equiv (\widehat{T}_1(\beta_1, \ldots, \beta_B), \ldots, \widehat{T}_N(\beta_1, \ldots, \beta_B)).$$

This function represents the output of the voting machine for any input deck of filled-out ballots $(\beta_1, \ldots, \beta_B)$. We say that the voting machine represented by the vector-valued function $\widehat{T}(\cdot)$ is not operating correctly if there exists a deck $(\beta_1, \ldots, \beta_B)$ and a candidate $i \in \mathcal{N}$ such that $\widehat{T}_i(\beta_1, \ldots, \beta_B) \neq T_i^*(\beta_1, \ldots, \beta_B)$. If the voting machine is not operating correctly, and if the voting machine is used to count votes in an actual election, then it could produce results inconsistent with the actual ballots cast and change the outcome of the election.

3.2 Formulation of RLAT

We now introduce the mathematical description of Robust Logic and Accuracy Testing (RLAT), an optimization-based framework for designing test decks in LAT with rigorous security guarantees. Specifically, given an uncertainty set $\mathcal{U}$ of possible ways that a voting machine might be operating incorrectly, RLAT designs the test deck by solving the following optimization problem:

$$\begin{aligned} \underset{B \in \mathbb{N},\ \beta_1, \ldots, \beta_B \in \mathscr{B}}{\text{minimize}} \quad & B \\ \text{subject to} \quad & \widehat{T}(\beta_1, \ldots, \beta_B) \neq T^*(\beta_1, \ldots, \beta_B) \quad \forall \widehat{T}(\cdot) \in \mathcal{U}. \end{aligned} \tag{RO}$$

The optimization problem (RO) yields a minimum-length test deck that is guaranteed to detect whether a voting machine is operating incorrectly in any of the ways specified by the uncertainty set.

In greater detail, the decision variables of the optimization problem (RO) consist of the length of the test deck, $B \in \mathbb{N}$, as well as the test deck of filled-out ballots without any overvotes, $\beta_1, \ldots, \beta_B \in \mathscr{B}$. The constraints of the optimization problem (RO) ensure that if the test deck is cast into a voting machine, and if the voting machine is operating incorrectly in any of the ways specified by the uncertainty set, then the output of the voting machine will be different from the output of a voting machine that is operating correctly. In other words, if $(B, \beta_1, \ldots, \beta_B)$ is an optimal solution for the optimization problem (RO), then we have a guarantee that the output of a voting machine $\widehat{T}(\beta_1, \ldots, \beta_B)$ will be different from the output of a voting machine that operates correctly $T^*(\beta_1, \ldots, \beta_B)$ whenever the voting machine $\widehat{T}(\cdot)$ is operating incorrectly in any of the ways specified by the uncertainty set $\mathcal{U}$. We elaborate on the construction of the uncertainty set in §3.3. The objective of the optimization problem (RO) is find a test deck that satisfies the constraints that consists of the fewest number of ballots.

The optimization problem (RO) for designing test decks in LAT can be viewed as attractive from the perspective of the relevant stakeholders including election administrators, policy makers, voters, and the computer security community. We elaborate below on the attractiveness and the design of the optimization problem (RO) through the perspectives of these various stakeholders:

Election Administrators. The administration of U.S. elections is a complicated endeavor, conducted in parallel by thousands of local officials across the country. Across so diverse and decentralized a system, even marginal increases to the difficulty or complexity of election procedures carry a very high administrative cost. This cautions against testing procedures that are substantially more difficult or resource intensive than those already in use.

From an implementation standpoint, RLAT aims to minimize the burden the solution confers to election administrators. By finding a test deck that minimizes the number of ballots, the optimization problem (RO) yields a suitable test deck that minimizes the time it takes to fill out and insert the decks into a machine. Moreover, a computer algorithm for solving the optimization problem (RO) can be integrated seamlessly at many stages, like in the vendor provided Election Management System (EMS) or by third-party ballot-printing companies that are often contracted to prepare test decks under current practice. This means we can implement RLAT through the operational processes that election administrators already have in place, with minimal change to the official's direct experience. Finally, using a computer algorithm can relieve election workers from the arduous task of manually designing test decks.

Policy Makers and Voters. RLAT is attractive for policy makers and voters because it enables LAT to provide strong and interpretable guarantees regarding the security of an election and the legitimacy of its outcome. Indeed, the optimization problem (RO) provides policy makers with the flexibility to specify the uncertainty set of possible ways that a voting machine may operate incorrectly. Policy makers can make this decision based on their evaluation of the cost-security trade-offs for their own state or jurisdiction, and based on factors like the known traits of the voting machines that are used in their elections (see Remark 2 in §3.3). The optimization problem (RO) can also easily integrate minimum legal requirements on test decks that are specified by policy makers, as we elaborate in Appendix D. Moreover, given a defined uncertainty set $\mathcal{U}$ and a test deck $(\beta_1, \ldots, \beta_B)$, any voter can verify that the test deck satisfies the constraints of the optimization problem (RO). This can enhance voter confidence that testing is being conducted fairly, and provides concrete and voter-verifiable assurances that the outcome of the election has not been accidentally or maliciously altered in ways similar to those from the examples given in §2.

Computer Security Community. In the election security community, and in the computer security community more broadly, risk is defined and minimized by considering a hypothetical adversary. This adversary aims to interfere with a system, and is constrained by a threat model which specifies the scope of their capabilities. This allows for the development of security interventions which have a definite effect with respect to certain assumptions about the options available to an adversary. The optimization problem (RO) thus works constructively with the computer security mindset. Indeed, the uncertainty set $\mathcal{U}$ is in essence a formalization of the threat model—it describes the potential modifications to the machine, which the adversary is able to choose between. By changing the uncertainty set, this formulation can flexibly substitute threat models as needed.

3.3 RLAT with the Swap Uncertainty Set

The key to achieving strong and interpretable security guarantees through RLAT is selecting an appropriate uncertainty set $\mathcal{U}$ in the optimization problem (RO). If the uncertainty set accounts for only a small number of possible misconfigurations or errors, then the security guarantees afforded by RLAT will be limited. On the other hand, if the uncertainty set is overly expansive, then (RO) might yield a test deck comprised of an impractically large number of ballots. Naturally, the task of choosing an uncertainty set that strikes an appropriate balance between the expressiveness and conservatism is a central challenge when constructing robust optimization problems such as (RO).

We focus throughout this paper on solving the optimization problem (RO) with a specific construction of the uncertainty set that we henceforth refer to as the swap uncertainty set. The swap uncertainty set consists of all of the voting machines that have an incorrect bijective mapping from candidates to voting targets. Hence, the optimization problem (RO) with the swap uncertainty set will yield the shortest test deck that is guaranteed to detect whether a voting machine is swapping votes across candidates. The optimization problem (RO) with the swap uncertainty set is stated formally at the end of the present §3.3 as the optimization problem (RO-Σ).

The swap uncertainty set is attractive from a security standpoint because it encompasses a general class of misconfigurations that would be difficult to detect for a well-implemented voting machine. The premise of the swap uncertainty set when scanning hand-marked ballots is that a voting machine is configured with a (x, y) coordinate for each candidate, which specifies the location of that candidate's voting target on the physical ballot. A well-implemented voting machine's software should perform two basic sanity checks of this configuration. First, the voting machine should not allow any candidate to be associated with multiple targets. Second, the voting machine should not allow different candidates' targets to overlap. This ensures a bijective mapping between candidates and targets. The premise of the swap uncertainty set when scanning ballots produced with a ballot-marking device (BMD) is that the BMD and optical scanner may be configured with inconsistent data representations of the candidates [12], such that votes encoded by the BMD as corresponding to one candidate may be read by the scanner as corresponding to another. Well-implemented software should enforce that each candidate has precisely one data representation, so this mismatch must also be a bijection. The swap uncertainty set is thus a natural choice for the RLAT problem under either of these models, since it describes each possible mapping from candidates to targets.

The formal definition of the swap uncertainty set requires the following additional notation. Let Σ denote the set of all non-identity bijections of the form $\sigma : \mathcal{N} \to \mathcal{N}$, where we say that the function $\sigma(\cdot)$ is a non-identity bijection if and only if the function satisfies the following two criteria:

1. For every target $j \in \mathcal{N}$, there exists one candidate $i \in \mathcal{N}$ that satisfies $\sigma(i) = j$.
2. There exists $i \in \mathcal{N}$ that satisfies $\sigma(i) \neq i$.

Each non-identity bijection can be understood as an incorrect mapping from candidates to targets.[4] The output of a voting machine whose mapping from candidates to targets is the bijection $\sigma : \mathcal{N} \to \mathcal{N}$ is given for each candidate $i \in \mathcal{N}_c$ in each contest $c \in \mathcal{C}$ by

$$T_i^\sigma(\beta_1, \ldots, \beta_B) \triangleq \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\right\}.$$

To make sense of the above definition, we remark that the inclusion $\sigma(i) \in \beta_b$ holds if and only if the b-th filled-out ballot in the test deck is interpreted by a voting machine with mapping σ to contain a vote for candidate i. Similarly, we observe that the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c$ holds if and only if the b-th filled-out ballot in the test deck is interpreted by a voting machine with mapping σ as containing votes for at most v_c candidates in contest c.

In view of the above notation, we define the swap uncertainty set as the set of the voting machines that correspond to each of the non-identity bijections:

$$\mathcal{U} \triangleq \{T^\sigma(\cdot) \equiv (T_1^\sigma(\cdot), \ldots, T_N^\sigma(\cdot)) : \sigma \in \Sigma\}.$$

Hence, we conclude that a test deck comprised of filled-out ballots $\beta_1, \ldots, \beta_B \in \mathscr{B}$ will satisfy the constraints of the optimization problem (RO) with the swap uncertainty set if and only if the test deck is guaranteed to detect whether a voting machine has been misconfigured to swap votes across candidates.

Equipped with the swap uncertainty set, we are ready to formally state the key optimization problem of this paper, that is, the optimization problem of finding a minimum-length test deck that is guaranteed to detect whether a voting machine has been misconfigured to swap votes across candidates. This optimization problem (RO) with the swap uncertainty set is stated below as (RO-Σ):

$$\begin{aligned} \underset{B \in \mathbb{N},\, \beta_1, \ldots, \beta_B \in \mathscr{B}}{\text{minimize}} \quad & B \\ \text{subject to} \quad & T^\sigma(\beta_1, \ldots, \beta_B) \neq T^*(\beta_1, \ldots, \beta_B) \quad \forall \sigma \in \Sigma. \end{aligned} \tag{RO-Σ}$$

[4]We note that the voting machine that operates correctly can be represented by the identity function $* : \mathcal{N} \to \mathcal{N}$, defined as the function that satisfies the equality $*(i) = i$ for all $i \in \mathcal{N}$.

Having established that the test decks obtained by (RO-Σ) offer attractive and rigorous security guarantees, we show in the rest of this paper that (RO-Σ) leads to test decks that can be practically deployed in real world elections. In §4 and §5, we develop an exact algorithm for solving the optimization problem (RO-Σ). In §6, we show that our exact algorithm scales to Michigan's November 2022 elections and that the test decks obtained by (RO-Σ) in those elections are not much longer than the test decks produced according to the heuristics Michigan currently uses. Hence, RLAT with the swap uncertainty set strikes a balance between producing test decks that account for a large number of possible voting machine misconfigurations and producing test decks with a practically small number of ballots.

Remark 2. Although this paper focuses on solving (RO-Σ), we note that RLAT offers election officials the flexibility to use uncertainty sets that include a more expansive or narrow model of the ways in which a voting machine could be wrong. For instance, in states that currently use weaker heuristics than Michigan's to prepare their test decks, election officials may be accustomed to using very short test decks and thus might balk at the lengths of test decks produced by (RO-Σ). To accommodate election officials in such states, one can solve (RO) with an uncertainty set that is a subset of the swap uncertainty set to obtain shorter test decks with weaker, albeit still rigorous defined, security guarantees (e.g. by opting to ignore the possibility of swaps between candidates in noncompetitive contests). Conversely, the swap uncertainty set can be made more expansive (e.g. by considering cases where the mapping of targets to candidates need not be bijective for voting machines whose software implementation allows the same target to be associated with multiple candidates, or vice versa). That being said, we emphasize that the algorithms presented in this paper are designed for solving (RO-Σ), i.e., the specific case of (RO) in which the uncertainty set is the swap uncertainty set.

3.4 Discussion

We conclude §3 by characterizing the key structural properties of test decks that satisfy the constraints of the optimization problem (RO-Σ). Specifically, the main contribution of §3.4 is a technical result, denoted below by Theorem 1, that characterizes the situations in which the output of a voting machine that operates correctly will be different from the output of a voting machine whose mapping from candidates to targets is a non-identity bijection. The characterization established by the following theorem will be used extensively for designing algorithms in the rest of the sections.

Theorem 1. *Let $\beta_1, \ldots, \beta_B \in \mathscr{B}$ and $\sigma \in \Sigma$. Then $T^\sigma(\beta_1, \ldots, \beta_B) \neq T^*(\beta_1, \ldots, \beta_B)$ if and only if at least one of the following two conditions hold:*

- *There exists a candidate $i \in \mathcal{N}$ that satisfies*

$$|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \neq |\{b \in \{1, \ldots, B\} : \sigma(i) \in \beta_b\}|.$$

- *There exist a contest $c \in \mathcal{C}$ and a filled-out ballot β_b for some $b \in \{1, \ldots, B\}$ that satisfy*

$$|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| > v_c.$$

The proof of this theorem and all other technical proofs in this paper can be found in Appendix F.

In words, the above theorem establishes that the output of a voting machine that operates correctly will not equal the output of a voting machine whose mapping from candidates to targets is a non-identity bijection $\sigma \in \Sigma$ if and only if the test deck comprised of filled-out ballots $\beta_1, \ldots, \beta_B \in$

$\mathscr{B}$ satisfies at least one of two conditions. The first condition is that there exists a candidate $i \in \mathcal{N}$ such that the number of filled-out ballots that selected target i, $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}|$, is different from the number of filled-out ballots that selected target $\sigma(i)$, $|\{b \in \{1, \ldots, B\} : \sigma(i) \in \beta_b\}|$. The second condition is that there exists a contest $c \in \mathcal{C}$ in one of the ballots $b \in \{1, \ldots, B\}$ that is interpreted as overvoted by the voting machine whose mapping from candidates to targets is σ. As an immediate corollary of Theorem 1, we obtain the following characterization of the test decks that are feasible for the optimization problem (RO-Σ).

Corollary 1. *A tuple $(B, \beta_1, \ldots, \beta_B)$ is feasible for the optimization problem (RO-Σ) if and only if $B \in \mathbb{N}$, $\beta_1, \ldots, \beta_B \in \mathscr{B}$, and for every $\sigma \in \Sigma$, at least one of the following two conditions hold:*

- *There exists a candidate $i \in \mathcal{N}$ that satisfies*

$$|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \neq |\{b \in \{1, \ldots, B\} : \sigma(i) \in \beta_b\}|.$$

- *There exist a contest $c \in \mathcal{C}$ and a filled-out ballot β_b for some $b \in \{1, \ldots, B\}$ that satisfy*

$$|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| > v_c.$$

Corollary 1 implies that the optimization problem (RO-Σ) always has a feasible solution. Specifically, it follows from Corollary 1 that any test deck that gives a distinct total number of votes to each of the candidates across all of the contests is a feasible solution for the optimization problem (RO-Σ) (a formal proof of this can be found in the proof of Proposition 4 from Appendix A). We note that while the Corollary 1 implies that a feasible solution for optimization problem (RO-Σ) can be obtained by simply giving every candidate across every contest a distinct number of votes, we show in Appendix A using real-world data that heuristics based on assigning a distinct number of votes to each candidate will result in test decks that contain too many ballots to be implementable in practice. Thus motivated, we proceed in §4 to develop an exact algorithm which solves the optimization problem (RO-Σ) in order to find test decks that are feasible solutions for (RO-Σ) with the fewest possible number of ballots.

4 Exact Algorithm

In this section, we present our exact algorithm for solving the optimization problem (RO-Σ).

4.1 Overview of Exact Algorithm

To begin our discussion of our exact algorithm, we recall from §1.1 that one of the key challenges in solving the optimization problem (RO-Σ) is that the problem contains an enormous number of constraints. Indeed, we observe that the number of constraints in the optimization problem (RO-Σ) is driven by the cardinality of the set of non-identity bijections Σ, and it follows readily from §3.3 that the number of non-identity bijections satisfies $|\Sigma| = N! - 1$ for a ballot style with N candidates. Because the number of candidates in real-world ballot styles often satisfies $N \geq 100$, an optimization problem that explicitly encodes all possible non-identity bijections thus cannot be represented nor solved on any extant computer for many real-world elections. It is currently unknown whether (RO-Σ) is NP-hard, and it is unknown whether there exists a mixed-integer linear programming reformulation of (RO-Σ) of size that is polynomial in the number of candidates N.

To contend with the computational challenge of solving the optimization problem (RO-Σ), we draw inspiration from an algorithmic strategy known as the cutting plane method [16, 13]. The

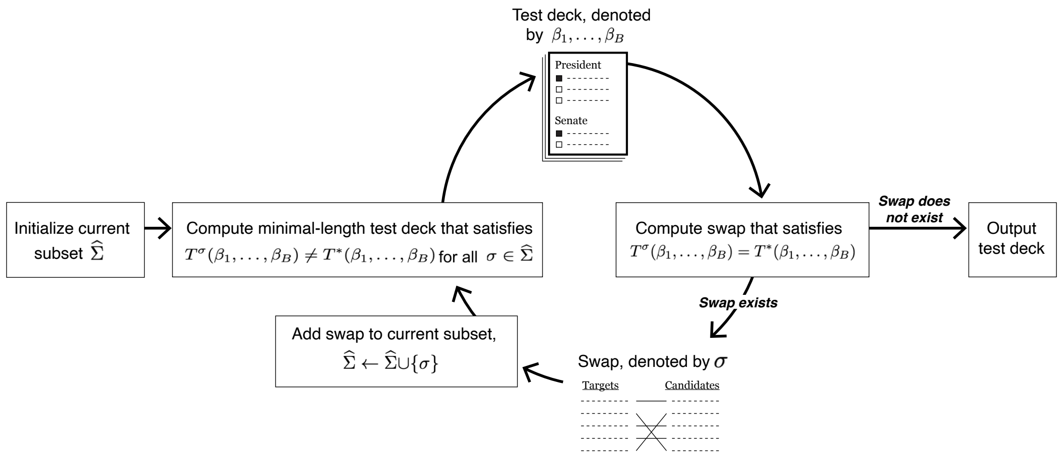

Test deck, denoted by $\beta_1, \dots, \beta_B$
President
Senate
Initialize current subset $\widehat{\Sigma}$
Compute minimal-length test deck that satisfies $T^\sigma(\beta_1, \dots, \beta_B) \neq T^*(\beta_1, \dots, \beta_B)$ for all $\sigma \in \widehat{\Sigma}$
Compute swap that satisfies $T^\sigma(\beta_1, \dots, \beta_B) = T^*(\beta_1, \dots, \beta_B)$
Swap does not exist
Output test deck
Swap exists
Swap, denoted by σ
Targets
Candidates
Add swap to current subset, $\widehat{\Sigma} \leftarrow \widehat{\Sigma} \cup \{\sigma\}$

Figure 5: Visualization of our exact algorithm from §4.1 for solving the optimization problem (RO-Σ).

goal of the cutting plane method is to circumvent the need to solve an optimization problem with a large number of constraints by solving a sequence of optimization problems with small numbers of constraints. The application of the cutting plane method to the optimization problem (RO-Σ) takes the form of an iterative algorithm that is described below and visualized in Figure 4.

In each iteration of our algorithm, we start with a subset of non-identity bijections $\widehat{\Sigma} \subseteq \Sigma$, and we solve the following variant of the optimization problem (RO-Σ):

$$\begin{aligned} \underset{B \in \mathbb{N},\, \beta_1, \dots, \beta_B \in \mathscr{B}}{\text{minimize}} \quad & B \\ \text{subject to} \quad & T^\sigma(\beta_1, \dots, \beta_B) \neq T^*(\beta_1, \dots, \beta_B) \quad \forall \sigma \in \widehat{\Sigma}. \end{aligned} \tag{RO-$\widehat{\Sigma}$}$$

To make sense of the optimization problem (RO-$\widehat{\Sigma}$), let us reflect on the relationship between (RO-$\widehat{\Sigma}$) and (RO-Σ). We observe that the optimization problem (RO-$\widehat{\Sigma}$) is nearly identical to the optimization problem (RO-Σ), with the only difference being that the former only has a constraint for each $\sigma \in \widehat{\Sigma}$ instead of a constraint for each $\sigma \in \Sigma$. The optimization problem (RO-$\widehat{\Sigma}$) can thus be viewed as a relaxation of the optimization problem (RO-Σ), in the sense that the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$) is less than or equal to the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$), but an optimal solution for the optimization problem (RO-$\widehat{\Sigma}$) might not be a feasible solution for the optimization problem (RO-Σ). The potential attractiveness of the optimization problem (RO-$\widehat{\Sigma}$) can be attributed to practical tractability: if the cardinality of $\widehat{\Sigma}$ is significantly less than the cardinality of Σ, then it will be possible to solve the optimization problem (RO-$\widehat{\Sigma}$) much faster by a computer compared to the optimization problem (RO-Σ).

After computing an optimal solution for the optimization problem (RO-$\widehat{\Sigma}$), the next step of the current iteration of our algorithm is determining whether the optimal solution for the optimization problem (RO-$\widehat{\Sigma}$) is a feasible solution for the optimization problem (RO-Σ). This step is performed by solving the optimization problem

$$\begin{aligned} \underset{\sigma \in \Sigma}{\text{minimize}} \quad & 0 \\ \text{subject to} \quad & T^\sigma(\beta_1, \dots, \beta_B) = T^*(\beta_1, \dots, \beta_B), \end{aligned} \tag{CUT}$$

where $(B, \beta_1, \dots, \beta_B)$ denotes the optimal solution of the optimization problem (RO-$\widehat{\Sigma}$). The optimization problem (CUT) has two possible outputs. First, if the optimization problem (CUT) outputs an optimal solution $\sigma \in \Sigma$, then we conclude that $(B, \beta_1, \dots, \beta_B)$ is not a feasible solution of the optimization problem (RO-Σ), since the constraint $T^\sigma(\beta_1, \ldots, \beta_B) \neq T^*(\beta_1, \ldots, \beta_B)$ in the optimization problem (RO-Σ) is violated by the test deck $(\beta_1, \ldots, \beta_B)$. Second, if the optimization problem (CUT) does not have any optimal solution, then we conclude that $(B, \beta_1, \ldots, \beta_B)$ is a feasible solution for the optimization problem (RO-Σ).

The final step of each iteration of the algorithm depends on the output of the optimization problem (CUT). If that optimization problem does not have a feasible solution, then we observe that $(B, \beta_1, \ldots, \beta_B)$ must be a feasible solution for the optimization problem (RO-Σ). Moreover, since $(B, \beta_1, \ldots, \beta_B)$ was an optimal solution for the optimization problem (RO-$\widehat{\Sigma}$), and since the optimization problem (RO-$\widehat{\Sigma}$) is a relaxation of the optimization problem (RO-Σ), it must be the case that $(B, \beta_1, \ldots, \beta_B)$ is also an optimal solution for the optimization problem (RO-Σ). Hence, if the optimization problem (CUT) does not have a feasible solution, then we have found an optimal solution to the optimization problem (RO-Σ), and the algorithm terminates. Otherwise, if the optimization problem (CUT) outputs $\sigma \in \Sigma$, then we conclude the current iteration by updating $\widehat{\Sigma} \leftarrow \widehat{\Sigma} \cup \{\sigma\}$ and starting a new iteration of the algorithm.

It follows from straightforward arguments that the algorithm described above will terminate at an optimal solution for the optimization problem (RO-Σ) after finitely many iterations, regardless of the choice of the subset $\widehat{\Sigma} \subseteq \Sigma$ in the first iteration. Indeed, the finite convergence of the algorithm follows from the fact that Σ is a finite set and from the fact that the optimization problem (CUT) will never output a non-identity bijection $\sigma \in \Sigma$ that is an element of $\widehat{\Sigma}$ when the test deck $(\beta_1, \ldots, \beta_B)$ satisfies the constraints of the optimization problem (RO-$\widehat{\Sigma}$). Therefore, the number of iterations of the algorithm is always upper bounded by $|\Sigma| = N! - 1$. In our numerical experiments throughout this paper, we initialize $\widehat{\Sigma}$ in the first iteration to be the empty set.

In order for the algorithm described above to be practically efficient in real-world elections, three important properties must hold. First, it must be possible to quickly solve the optimization problem (RO-$\widehat{\Sigma}$) when $|\widehat{\Sigma}| \ll |\Sigma|$. Second, the algorithm must terminate after a relatively small number of iterations, as this property is essential for ensuring that the cardinality of $\widehat{\Sigma}$ remains much smaller than the cardinality of Σ. Third, it must be possible to have a fast implementation of the optimization problem (CUT) for finding a constraint that is violated by the test deck obtained by solving the optimization problem (RO-$\widehat{\Sigma}$). In the subsequent §4.2 and §5, we show that these three important properties for obtaining a practically efficient algorithm can be achieved simultaneously.

4.2 Mixed-Integer Reformulations

In each iteration of the cutting plane method from §4.1, we are tasked with solving the optimization problems (RO-$\widehat{\Sigma}$) and (CUT). Here, we show that optimal solutions for these two optimization problems can be obtained by solving mixed-integer linear optimization problems. In doing so, this subsection enables the cutting plane method from §4.1 to be easily implementable using widely available open-source and commercial optimization software such as Gurobi and Mosek. Improvements to the mixed-integer linear optimization reformulations from the present subsection are proposed and analyzed in the subsequent §5.

4.2.1 Mixed-Integer Reformulation of (RO-$\widehat{\Sigma}$)

At a high level, our procedure for solving the optimization problem (RO-$\widehat{\Sigma}$) consists of the following steps. First, we fix B to be an integer that is less than or equal to the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$). We then solve the following optimization problem:

$$\begin{aligned} \underset{\beta_1,\ldots,\beta_B \in \mathscr{B}}{\text{minimize}} \quad & 0 \\ \text{subject to} \quad & T^\sigma(\beta_1,\ldots,\beta_B) \neq T^*(\beta_1,\ldots,\beta_B) \quad \forall \sigma \in \widehat{\Sigma}. \end{aligned} \tag{RO-$\widehat{\Sigma}$-B}$$

If the optimization problem (RO-$\widehat{\Sigma}$-B) does not have any feasible solutions, then we observe that B must be strictly less than the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$). In that case, we update $B \leftarrow B+1$ and re-solve the optimization problem (RO-$\widehat{\Sigma}$-B) with the new value for the parameter B. We repeat this loop until the optimization problem (RO-$\widehat{\Sigma}$-B) is feasible, at which point (RO-$\widehat{\Sigma}$-B) will yield an optimal solution for the optimization problem (RO-$\widehat{\Sigma}$).

Remark 3. We use the above procedure to solve the optimization problem (RO-$\widehat{\Sigma}$) because the number of decision variables in the optimization problem (RO-$\widehat{\Sigma}$) depends on the integer B, where the integer B is itself a decision variable in the optimization problem (RO-$\widehat{\Sigma}$). In contrast, the number of decision variables in the optimization problem (RO-$\widehat{\Sigma}$-B) is known because the integer B is fixed externally. Because the number of decision variables is known a priori, the optimization problem (RO-$\widehat{\Sigma}$-B) can be reformulated as a mixed-integer linear optimization problem.

Remark 4. The above procedure requires B to be initialized to an integer that is less than or equal to the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$). In our implementation of the procedure, we initialize the integer to $B \leftarrow 1$ in the first iteration of the cutting plane method. In all subsequent iterations of the cutting plane method, we initialize B to the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$) from the previous iteration of the cutting plane method.[5]

In the remainder of §4.2.1, we show that the optimization problem (RO-$\widehat{\Sigma}$-B) can be reformulated as a mixed-integer linear optimization problem. Indeed, let $\mathcal{B} \triangleq \{1,\ldots,B\}$ and $\mathcal{B}_0 \triangleq \{0\} \cup \mathcal{B}$. With this notation, we first observe that the optimization problem (RO-$\widehat{\Sigma}$-B) can be rewritten equivalently as the following intermediary optimization problem:

$$\begin{aligned} \underset{\beta \in \{0,1\}^{\mathcal{B}\times\mathcal{N}}}{\text{minimize}} \quad & 0 && \text{(1a)} \\ \text{subject to} \quad & \sum_{i \in \mathcal{N}_c} \beta_{b,i} \le v_c \qquad \forall b \in \mathcal{B}, c \in \mathcal{C} && \text{(1b)} \\ & T^\sigma(\{i : \beta_{1,i} = 1\},\ldots,\{i : \beta_{B,i} = 1\}) \\ & \quad \neq T^*(\{i : \beta_{1,i} = 1\},\ldots,\{i : \beta_{B,i} = 1\}) \qquad \forall \sigma \in \widehat{\Sigma}. && (-) \end{aligned}$$

The optimization problem (1) can be interpreted as follows. Each binary decision variable $\beta_{b,i} \in \{0,1\}$ is equal to one if and only if target i is selected in the b-th filled-out ballot in the test deck. Hence, each vector $\beta_b \equiv (\beta_{b,1},\ldots,\beta_{b,N}) \in \{0,1\}^{\mathcal{N}}$ serves as a binary encoding of the targets that are selected in the b-th filled-out ballot. Constraint (1b) ensures that each filled-out ballot in the test deck is feasible, that is, there does not exist a filled-out ballot that contains more votes for candidates in a contest than are allowed. Constraint (–) says that the output of a voting machine

[5]We recall that $\widehat{\Sigma}$ in the current iteration of the cutting plane method is a superset of $\widehat{\Sigma}$ from the previous iteration of the cutting plane method. As a result, the constraints of the optimization problem (RO-$\widehat{\Sigma}$) in the current iteration of the cutting plane method is always a strict superset of the constraints of the optimization problem (RO-$\widehat{\Sigma}$) in the previous iteration of the cutting plane method. This implies that the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$) from the previous iteration of the cutting plane method is always less than or equal to the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$) in the current iteration of the cutting plane method.

whose mapping from candidates to targets is $\sigma \in \widehat{\Sigma}$ must be different from the output of a correctly operating voting machine when using the test deck $(\{i : \beta_{1,i} = 1\}, \ldots, \{i : \beta_{B,i} = 1\})$.

We observe from inspection that the above optimization problem (1) consists exclusively of binary decision variables. Moreover, the objective function (1a) and the constraints (1b) are linear functions of the decision variables. Therefore, the final step in our reformulation of the optimization problem (RO-$\widehat{\Sigma}$-B) as a mixed-integer linear optimization problem is to reformulate the constraint (–). This can be done through the introduction of new decision variables γ, y, and p, resulting in the following mixed-integer linear optimization problem:

$$
\begin{aligned}
&\underset{\substack{\beta \in \{0,1\}^{\mathcal{B}\times\mathcal{N}} \\ \gamma \in \{0,1\}^{\mathcal{N}\times\mathcal{B}_0},\, y \in \mathbb{R}_{\geq 0}^{\mathcal{N}\times\mathcal{N}} \\ p^{\sigma} \in \{0,1\}^{\mathcal{B}\times\mathcal{C}} \forall \sigma \in \widehat{\Sigma}}}{\text{minimize}} \quad && (1a) && \\
&\text{subject to} \quad && (1b) && \\
& && \sum_{g \in \mathcal{B}_0} \gamma_{i,g} = 1 && \forall i \in \mathcal{N} \qquad (1c)\\
& && \sum_{b \in \mathcal{B}} \beta_{b,i} = \sum_{g \in \mathcal{B}_0} g \gamma_{i,g} && \forall i \in \mathcal{N} \qquad (1d)\\
& && y_{i,j} \geq -1 + \gamma_{i,g} + \gamma_{j,g} && \forall i,j \in \mathcal{N}, g \in \mathcal{B}_0 \qquad (1e)\\
& && p^{\sigma}_{b,c} \geq 1 - \frac{1}{v_c + 1} \sum_{i \in \mathcal{N}_c} \beta_{b,\sigma(i)} && \forall \sigma \in \widehat{\Sigma}, b \in \mathcal{B}, c \in \mathcal{C} \qquad (1f)\\
& && \sum_{i \in \mathcal{N}} \left(1 - y_{i,\sigma(i)}\right) + \sum_{b \in \mathcal{B}} \sum_{c \in \mathcal{C}} \left(1 - p^{\sigma}_{b,c}\right) \geq 1 && \forall \sigma \in \widehat{\Sigma}. \qquad (1g)
\end{aligned}
$$

The decision variables of the above optimization problem can be interpreted as follows.

First, we observe that constraints (1c) and (1d) together enforce that each decision variable $\gamma_{i,g} \in \{0,1\}$ will be equal to one if and only if candidate i appears in exactly g ballots.

Second, constraint (1e) requires that $y_{i,j}$ must be greater than or equal to one if candidates i and j receive the same number of votes across $\mathcal{B}$, and may be as low as zero if they receive a different number of votes. Similarly, constraint (1f) requires that $p^{\sigma}_{b,c}$ must be equal to one if contest c receieves v_c or fewer votes on ballot b under swap σ, and may be zero if the contest is instead overvoted.

Finally, constraint (1g) ensures that a feasible solution for the optimization problem (1) exists if and only if at least one $y_{i,\sigma(i)}$ or $p^{\sigma}_{b,c}$ is zero for each $\sigma \in \widehat{\Sigma}$. In other words, the problem has a feasible solution if and only if, for each swap in $\widehat{\Sigma}$, there exists at least one candidate that is mapped to a target with a different number of votes *or* there exists at least one ballot that is unexpectedly interpreted as containing an overvote. This ensures that any feasible solution to this optimization problem corresponds to a deck of ballots which will detect every swap in our subset.

In summary, we have shown in the present §4.2.1 that the optimization problem (RO-$\widehat{\Sigma}$) can be solved by a procedure that consists of fixing the integer B to a lower bound on the optimal objective value of the optimization problem (RO-$\widehat{\Sigma}$) and then incrementing B until the optimization problem (RO-$\widehat{\Sigma}$-B) has a feasible solution. Moreover, for each fixed choice of the integer B, we showed that the optimization problem (RO-$\widehat{\Sigma}$-B) can be reformulated as the mixed-integer linear optimization problem (1). Thus, we have shown that an optimal solution for the optimization problem (RO-$\widehat{\Sigma}$) can be obtained by a procedure that consists of solving one or more mixed-integer linear optimization problems.

4.2.2 Mixed-Integer Reformulation of (CUT)

We conclude §4.2 by reformulating the optimization problem (CUT) as a mixed-integer linear optimization problem.[6] Given any test deck $(\beta_1, \ldots, \beta_B)$, our mixed-integer linear optimization reformulation of the optimization problem (CUT) is the following:

$$
\begin{aligned}
\underset{x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}}{\text{minimize}} \quad & 0 \\
\text{subject to} \quad & \sum_{j \in \mathcal{N}} x_{i,j} = 1 && \forall i \in \mathcal{N} && \text{(2a)} \\
& \sum_{i \in \mathcal{N}} x_{i,j} = 1 && \forall j \in \mathcal{N} && \text{(2b)} \\
& \sum_{i \in \mathcal{N}} x_{i,i} \le |\mathcal{N}| - 2 && && \text{(2c)} \\
& \sum_{i \in \mathcal{N}_c} \sum_{j \in \beta_b} x_{i,j} \le v_c && \forall b \in \mathcal{B}, c \in \mathcal{C} && \text{(2d)} \\
& x_{i,j} = 0 && \forall i,j \in \mathcal{N}: \; |\{b \in \mathcal{B} : i \in \beta_b\}| \neq |\{b \in \mathcal{B} : j \in \beta_b\}|. && \text{(2e)}
\end{aligned}
$$

The constraints (2a)-(2c) enforce that the decision variables $x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$ are a binary encoding of a non-identity bijection $\sigma \in \Sigma$. Indeed, constraints (2a) and (2b) ensure that each feasible solution of (2) can be transformed into a bijection $\sigma : \mathcal{N} \to \mathcal{N}$ using the rule that $\sigma(i) = j$ if and only if $x_{i,j} = 1$ for each $i, j \in \mathcal{N}$. Constraint (2c) enforces that there exists $i \in \mathcal{N}$ that satisfies $\sigma(i) \neq i$.

The last two constraints (2d) and (2e) enforce that the non-identity bijection $\sigma \in \Sigma$ corresponding to the decision variables $x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$ satisfies the equality $T^\sigma(\beta_1, \ldots, \beta_B) = T^*(\beta_1, \ldots, \beta_B)$. To see why this is the case, we first observe that constraint (2d) enforces that the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c$ holds for all contests $c \in \mathcal{C}$ and $b \in \mathcal{B}$. Moreover, constraint (2e) enforces that the equality $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = |\{b \in \{1, \ldots, B\} : \sigma(i) \in \beta_b\}|$ holds for all candidates $i \in \mathcal{N}$. Therefore, it follows from Theorem 1 in §3.4 that constraints (2d) and (2e) are satisfied if and only if the non-identity bijection $\sigma \in \Sigma$ corresponding to the decision variables $x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$ satisfies the equality $T^\sigma(\beta_1, \ldots, \beta_B) = T^*(\beta_1, \ldots, \beta_B)$.

5 Improvements to Exact Algorithm

In this section, we present five improvements to the mixed-integer linear optimization reformulations from §4.2 that significantly increase the practical efficiency of the cutting plane method from §4.1. Our five improvements to the mixed-integer linear optimization reformulations are presented and analyzed in the subsequent §5.1-§5.5. In Appendix C, we demonstrate via numerical experiments on synthetic elections that each of the five improvements from this section, when applied in isolation, generates between a 20x to 3000x speedup to the cutting plane method.

5.1 Improvement 1: Reducing Number of Decision Variables and Constraints

As our first step in increasing the practical efficiency of the cutting plane method from §4.1, we show that a number of the decision variables and constraints in the mixed-integer linear optimization problem (1) from §4.2.1 can be removed without any loss of generality. By removing these unnecessary decision variables and constraints from the mixed-integer linear optimization problem (1),

[6] More precisely, our reformulation (2) of the optimization problem (CUT) is a *binary* linear optimization problem.

we demonstrate through numerical experiments in Appendix C that the computation time of each iteration of the cutting plane method can be significantly decreased.

To motivate our subsequent developments in §5.1, we begin by analyzing the size of the mixed-integer linear optimization problem (1). Indeed, we observe that the number of binary decision variables, the number of continuous decision variables, and the number of constraints in the mixed-integer linear optimization problem (1) are as follows:

$$\begin{aligned}
\#\text{ binary decision variables} &= \underbrace{(|\mathcal{B}| \times |\mathcal{N}|)}_{\beta} + \underbrace{(|\mathcal{B}_0| \times |\mathcal{N}|)}_{\gamma} + \underbrace{\left(|\mathcal{B}| \times |\mathcal{C}| \times |\widehat{\Sigma}|\right)}_{p} \\
&= BN + (B+1)N + BC|\widehat{\Sigma}| \\
&= \mathcal{O}\left(BN + BC|\widehat{\Sigma}|\right);
\end{aligned}$$

$$\begin{aligned}
\#\text{ continuous decision variables} &= \underbrace{|\mathcal{N}| \times |\mathcal{N}|}_{y} \\
&= N^2;
\end{aligned}$$

$$\begin{aligned}
\#\text{ constraints} &= \underbrace{(|\mathcal{B}| \times |\mathcal{C}|)}_{\text{(1b)}} + \underbrace{|\mathcal{N}|}_{\text{(1c)}} + \underbrace{|\mathcal{N}|}_{\text{(1d)}} + \underbrace{(|\mathcal{N}| \times |\mathcal{N}| \times |\mathcal{B}_0|)}_{\text{(1e)}} + \underbrace{\left(|\mathcal{B}| \times |\mathcal{C}| \times \left|\widehat{\Sigma}\right|\right)}_{\text{(1f)}} + \underbrace{|\widehat{\Sigma}|}_{\text{(1g)}} \\
&= BC + 2N + N^2(B+1) + BC|\widehat{\Sigma}| + |\widehat{\Sigma}| \\
&= \mathcal{O}\left(BN^2 + BC|\widehat{\Sigma}|\right).
\end{aligned}$$

In real-world elections such as those from Michigan, the number of contests typically satisfies $15 \le C \le 40$ (see Figure 3 in §3.1), the number of candidates typically satisfies $60 \le N \le 120$ (see Figure 3 in §3.1), the number of ballots in an optimal test deck for (RO-Σ) typically satisfies $20 \le B \le 50$ (see Figure 6a in §6.1), and the number of iterations of our cutting plane method typically satisfies $50 \le |\widehat{\Sigma}| \le 300$ (see Figure 7b in §6.2). Combining the equations derived above with the real-world data observed from Michigan, we conclude that the size of the mixed-integer linear optimization problem (1) is driven primarily by the binary decision variables p, the continuous decision variables y, and the constraints (1e) and (1f).

In view of the above motivation, we first show that a number of the binary decision variables $p^{\sigma}_{b,c} \in \{0,1\}$ and constraints (1f) can be removed from the mixed-integer linear optimization problem (1) without loss of generality. Indeed, we recall from the discussion in §4.2.1 that there always exists an optimal solution of the mixed-integer linear optimization problem (1) in which each binary decision variable $p^{\sigma}_{b,c}$ satisfies the equality $p^{\sigma}_{b,c} = 0$ if and only if contest c in ballot b is interpreted as containing an overvote by the voting machine whose mapping is σ. To decrease the number of these binary decision variables, we utilize the following intermediary result:

Lemma 1. *Let $\sigma \in \widehat{\Sigma}$ and $c \in \mathcal{C}$. If the inequality $\sum_{c' \in \mathcal{C}} \min\{|\{\sigma(i) \in \mathcal{N}_{c'} : i \in \mathcal{N}_c\}|, v_{c'}\} \le v_c$ holds, then every feasible solution of the mixed-integer linear optimization problem (1) satisfies the equality $p^{\sigma}_{b,c} = 1$ for all $b \in \mathcal{B}$.*

To make sense of the above lemma, we remark that $\sum_{c' \in \mathcal{C}} \min\{|\{\sigma(i) \in \mathcal{N}_{c'} : i \in \mathcal{N}_c\}|, v_{c'}\}$ is equal to the maximum number of votes that may be mapped to contest c under mapping σ for any filled-out ballot in $\mathscr{B}$. If this summation is less than or equal to v_c, then contest c will never be overvoted by a filled-out ballot from $\mathscr{B}$ under mapping σ.

In view of Lemma 1, we now demonstrate that a subset of the binary decision variables of the form $p^{\sigma}_{b,c}$ and a subset of the constraints (1f) can be removed from the mixed-integer linear optimization problem (1) without loss of generality. Indeed, for each non-identity bijection $\sigma \in \widehat{\Sigma}$, let the subset of contests that have the possibility of being overvoted under a voting machine with mapping σ be denoted by

$$\widehat{\mathcal{C}}^{\sigma} \triangleq \left\{ c \in \mathcal{C} : \sum_{c' \in \mathcal{C}} \min \left\{ \left| \left\{ \sigma(i) \in \mathcal{N}_{c'} : i \in \mathcal{N}_c \right\} \right|, v_{c'} \right\} \geq v_c + 1 \right\}.$$

We observe that the subset of contests $\widehat{\mathcal{C}}^{\sigma}$ for each $\sigma \in \widehat{\Sigma}$ can be efficiently precomputed.[7] Using these subsets of contests, it follows immediately from Lemma 1 that constraints (1f) and (1g) can without loss of generality be replaced by the following constraints:

$$p^{\sigma}_{b,c} \geq 1 - \frac{1}{v_c + 1} \sum_{i \in \mathcal{N}_c} \beta_{b,\sigma(i)} \qquad \forall \sigma \in \widehat{\Sigma}, b \in \mathcal{B}, c \in \widehat{\mathcal{C}}^{\sigma} \tag{3a}$$

$$\sum_{i \in \mathcal{N}} \left(1 - y_{i,\sigma(i)}\right) + \sum_{b \in \mathcal{B}} \sum_{c \in \widehat{\mathcal{C}}^{\sigma}} \left(1 - p^{\sigma}_{b,c}\right) \geq 1 \qquad \forall \sigma \in \widehat{\Sigma}. \tag{3b}$$

In particular, we observe that the binary decision variable $p^{\sigma}_{b,c} \in \{0,1\}$ for each $\sigma \in \widehat{\Sigma}$, $b \in \mathcal{B}$, and $c \in \mathcal{C}$ that satisfies $c \notin \widehat{\mathcal{C}}^{\sigma}$ no longer appears in the mixed-integer linear optimization problem (1) and can thus be eliminated.

Next, we show that a number of the continuous decision variables $y_{i,j}$ and constraints (1e) can be removed from the mixed-integer linear optimization problem (1) without loss of generality. Indeed, we recall from the discussion in §4.2.1 that each decision variable $y_{i,j} \in \mathbb{R}_{\geq 0}$ will at optimality be equal to zero only if candidates $i, j \in \mathcal{N}$ do not appear in the same number of ballots. Moreover, we observe that variable $y_{i,j}$ is only referenced in the constraint (1g) by the terms $y_{i,\sigma(i)}$ for each $i \in \mathcal{N}$ and $\sigma \in \widehat{\Sigma}$. Therefore, we observe that the decision variable $y_{i,j}$ only needs to be defined for the pairs of candidates (i,j) in the set

$$\mathcal{P}(\widehat{\Sigma}) \triangleq \left\{ (i,j) \in \mathcal{N}^2 : i \neq j \text{ and there exists } \sigma \in \widehat{\Sigma} \text{ that satisfies } \sigma(i) = j \right\},$$

and we can replace the constraint (1e) with

$$y_{i,j} \geq -1 + \gamma_{i,g} + \gamma_{j,g} \qquad \forall (i,j) \in \mathcal{P}(\widehat{\Sigma}), g \in \mathcal{B}_0. \tag{3c}$$

We conclude that the mixed-integer linear optimization problem (1) can be reduced to an optimization problem with the following number of binary decision variables, number of continuous

[7] By *precomputed*, we mean that the set $\widehat{\mathcal{C}}^{\sigma}$ can be computed independent of B and only needs to be computed once per σ. Hence, it suffices to compute $\widehat{\mathcal{C}}^{\sigma}$ when σ is first added by the cutting plane method into the set $\widehat{\Sigma}$. Moreover, the set $\widehat{\mathcal{C}}^{\sigma}$ can be computed in $\mathcal{O}(C^2 + N)$ time by the following straightforward algorithm: (1) initialize an $C \times C$-dimension array of all zeros; (2) for each $i \in \mathcal{N}$, increment the value in the array at position (c, c') if $i \in \mathcal{N}_c$ and $\sigma(i) \in \mathcal{N}_{c'}$; (3) for each $c \in \mathcal{C}$, calculate the quantity $\sum_{c' \in \mathcal{C}} \min \left\{ \left| \left\{ \sigma(i) \in \mathcal{N}_{c'} : i \in \mathcal{N}_c \right\} \right|, v_{c'} \right\}$ by summing the minimum of the value of the array at position (c, c') and $v_{c'}$ over all $c' \in \mathcal{C}$.

decision variables, and number of constraints:

$$\begin{aligned}
\#\text{ binary decision variables} &= \underbrace{(|\mathcal{B}| \times |\mathcal{N}|)}_{\beta} + \underbrace{(|\mathcal{B}_0| \times |\mathcal{N}|)}_{\gamma} + \underbrace{\left(|\mathcal{B}| \times \sum_{\sigma \in \widehat{\Sigma}} \left| \widehat{\mathcal{C}}^\sigma \right| \right)}_{p} \\
&= BN + (B+1)N + B \sum_{\sigma \in \widehat{\Sigma}} \left| \widehat{\mathcal{C}}^\sigma \right| \\
&= \mathcal{O}\left(BN + B \sum_{\sigma \in \widehat{\Sigma}} \left| \widehat{\mathcal{C}}^\sigma \right| \right);
\end{aligned}$$

$$\#\text{ continuous decision variables} = \underbrace{|\mathcal{P}(\widehat{\Sigma})|}_{y};$$

$$\begin{aligned}
\#\text{ constraints} &= \underbrace{(|\mathcal{B}| \times |\mathcal{C}|)}_{(1b)} + \underbrace{|\mathcal{N}|}_{(1c)} + \underbrace{|\mathcal{N}|}_{(1d)} + \underbrace{\left(|\mathcal{B}| \times \sum_{\sigma \in \widehat{\Sigma}} \left| \widehat{\mathcal{C}}^\sigma \right| \right)}_{(3a)} + \underbrace{|\widehat{\Sigma}|}_{(3b)} + \underbrace{\left(|\mathcal{P}(\widehat{\Sigma})| \times |\mathcal{B}_0| \right)}_{(3c)} \\
&= BN + BC + 2N + B \sum_{\sigma \in \widehat{\Sigma}} \left| \widehat{\mathcal{C}}^\sigma \right| + |\widehat{\Sigma}| + |\mathcal{P}(\widehat{\Sigma})|(B+1) \\
&= \mathcal{O}\left(B|\mathcal{P}(\widehat{\Sigma})| + B \sum_{\sigma \in \widehat{\Sigma}} \left| \widehat{\mathcal{C}}^\sigma \right| \right).
\end{aligned}$$

As we show in Appendix C, the above reductions in the number of decision variables and constraints lead to a significant decrease in the computation time for solving the mixed-integer linear optimization problem (1) in each iteration of the cutting plane method.

5.2 Improvement 2: Distinct Votes for Candidates in the Same Contest

As our second step in increasing the practical efficiency of the cutting plane method, we add a set of extra constraints into the mixed-integer linear optimization problem (1). These extra constraints force the mixed-integer linear optimization problem (1) to output a test deck that proactively satisfies many of the constraints $\sigma \in \Sigma$ from the optimization problem (RO-Σ) that were not explicitly included in subset $\widehat{\Sigma}$. As we demonstrate through numerical experiments in Appendix C, the addition of this set of extra constraints leads to a significant decrease in the number of iterations of the cutting plane method without any meaningful increase in the computation time for solving the mixed-integer linear optimization problem (1) in each iteration.

The motivation for the set of extra constraints is given by the following Lemma 2 and Proposition 1. In Lemma 2, we establish a structural property that is satisfied by every test deck that satisfies the constraints of the optimization problem (RO-Σ). Specifically, the following lemma shows that a test deck satisfies the constraints of the optimization problem (RO-Σ) only if all of the candidates that appear in the same contest receive a different number of votes, where we say that two candidates $i, j \in \mathcal{N}$ appear in the same contest if there exists a contest $c \in \mathcal{C}$ that satisfies $i, j \in \mathcal{N}_c$.

Lemma 2. *If* $(B, \beta_1, \ldots, \beta_B)$ *is a feasible solution for the optimization problem* (RO-Σ)*, then* $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \neq |\{b \in \{1, \ldots, B\} : j \in \beta_b\}|$ *for all candidates* $i < j$ *that appear in the same contest.*

Equipped with the above intermediary lemma, we show in the following Proposition 1 that there always exists an optimal solution for the optimization problem (RO-Σ) in which all of the candidates that appear in the same contest have a strictly increasing number of votes.

Proposition 1. *There exists an optimal solution for the optimization problem* (RO-Σ) *that satisfies* $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| < |\{b \in \{1, \ldots, B\} : j \in \beta_b\}|$ *for all candidates* $i < j$ *that appear in the same contest.*

Hence, the above proposition implies that we can, without loss of generality, restrict the solution spaces of the optimization problems (RO-Σ) and (RO-$\widehat{\Sigma}$) by adding extra constraints which enforce that $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| < |\{b \in \{1, \ldots, B\} : j \in \beta_b\}|$ for all candidates $i < j$ that appear in the same contest.

Motivated by the structure of optimal solutions for the optimization problem (RO-Σ) that is established by Proposition 1, we now describe the set of extra constraints that we add into the mixed-integer linear optimization problem (1). The purpose of this set of extra constraints is to ensure that every feasible solution (β, γ, y, p) of the mixed-integer linear optimization problem (1) satisfies the inequality

$$|\{b \in \{1, \ldots, B\} : \beta_{b,i} = 1\}| < |\{b \in \{1, \ldots, B\} : \beta_{b,j} = 1\}|$$

for all candidates $i < j$ that appear in the same contest. We accomplish this by adding the following set of extra constraints (4) into the mixed-integer linear optimization problem (1). In the following extra constraints, we use the shorthand notation $\mathcal{N}_c^k$ to denote the candidate with the kth smallest index among all candidates in the cth contest (with the indexing of candidates that appear in the same contest starting at index 1).

$$\sum_{b \in \mathcal{B}} \left(\beta_{b,\mathcal{N}_c^{k+1}} - \beta_{b,\mathcal{N}_c^k} \right) \geq 1 \quad \forall c \in \mathcal{C} \text{ and } k \in \{1, \ldots, |\mathcal{N}_c| - 1\}. \tag{4}$$

Indeed, we observe that the set of extra constraints (4) ensure that the number of votes for each candidate i is strictly less than the number of votes for candidate $j > i$ whenever candidates i and j appear in the same contest.

5.3 Improvement 3: Distinct Votes for Candidates in Similar Contests

As our third step in increasing the practical efficiency of the cutting plane method, we add a second set of extra constraints into the mixed-integer linear optimization problem (1). Similarly as §5.2, the set of extra constraints from the present §5.3 force the mixed-integer linear optimization problem (1) to output a test deck that proactively satisfies many of the constraints $\sigma \in \Sigma$ from the optimization problem (RO-Σ) that were not explicitly included in subset $\widehat{\Sigma}$. As we demonstrate through numerical experiments in Appendix C, the addition of this second set of extra constraints leads to a significant decrease in the number of iterations of the cutting plane method without any meaningful increase in the computation time for solving the mixed-integer linear optimization problem (1) in each iteration.

To describe our second set of extra constraints, we require some additional terminology. We begin with the following Definition 1, which provides a way of referring to contests that are similar to one another.

Definition 1 (Equivalence of contests)**.** We say that two contests $c, c' \in \mathcal{C}$ are equivalent, denoted by $c \equiv c'$, if and only if they satisfy $|\mathcal{N}_c| = |\mathcal{N}_{c'}|$ and $v_c = v_{c'}$.

In other words, we say that two contests are equivalent if and only if the contests have the same number of candidates and the same maximum number of votes. Next, recall from §5.2 that $\mathcal{N}_c^k$ refers to the candidate with the kth smallest index among all candidates in contest c. Equipped with this notation, the second additional terminology, which is denoted below by Definition 2, provides a way to compare the votes received by candidates in two equivalent contests.

Definition 2 (Lexicographic ordering of contests)**.** We say that two contests $c, c' \in \mathcal{C}$ are lexicographically ordered with respect to a test deck $\beta_1, \ldots, \beta_B \in \mathscr{B}$, denoted by $c \prec_{\beta_1 \cdots \beta_B} c'$, if and only if $c \equiv c'$ and there exists $k \in \{1, \ldots, |\mathcal{N}_c|\}$ that satisfies

$$\begin{gathered}
\left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_c^{|\mathcal{N}_c|} \in \beta_b\right\}\right| = \left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_{c'}^{|\mathcal{N}_c|} \in \beta_b\right\}\right| \\
\vdots \\
\left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_c^{k+1} \in \beta_b\right\}\right| = \left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_{c'}^{k+1} \in \beta_b\right\}\right| \\
\left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_c^{k} \in \beta_b\right\}\right| < \left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_{c'}^{k} \in \beta_b\right\}\right|.
\end{gathered}$$

In other words, we say that two contests are lexicographically ordered with respect to a test deck if and only if the contests are equivalent and the number of votes received by candidates in the first contest is lexicographically less than the number of votes received by candidates in the second contest, beginning with the highest-indexed candidates in the contests.

In view of the additional terminology given by Definitions 1 and 2, we now describe the second set of extra constraints that we add into the mixed-integer linear optimization problem (1). The motivation for this second set of extra constraints is given by the following Lemma 3 and Proposition 2. In Lemma 3, we establish a structural property that is satisfied by every test deck that satisfies the constraints of the optimization problem (RO-Σ). Specifically, the following lemma shows that a test deck satisfies the constraints of the optimization problem (RO-Σ) only if all equivalent contests are lexicographically distinct.

Lemma 3. *If $(B, \beta_1, \ldots, \beta_B)$ is a feasible solution for the optimization problem (RO-Σ), then $c \prec_{\beta_1 \cdots \beta_B} c'$ or $c' \prec_{\beta_1 \cdots \beta_B} c$ for all contests $c < c'$ that satisfy $c \equiv c'$.*

Equipped with the above intermediary lemma, we show in the following Proposition 2 that there always exists an optimal solution for the optimization problem (RO-Σ) in which all of the equivalent contests are lexicographically ordered according to the indices of the contests.

Proposition 2. *There exists an optimal solution for the optimization problem (RO-Σ) that satisfies $c \prec_{\beta_1 \cdots \beta_B} c'$ for all contests $c < c'$ that satisfy $c \equiv c'$.*

Hence, the above proposition implies that we can, without loss of generality, restrict the solution spaces of the optimization problems (RO-Σ) and (RO-$\widehat{\Sigma}$) by adding extra constraints which enforce that $c \prec_{\beta_1 \cdots \beta_B} c'$ for all contests $c < c'$ that satisfy $c \equiv c'$.

Motivated by the structure of optimal solutions for the optimization problem (RO-Σ) that is established by Proposition 2, we now describe the second set of extra constraints that we add into the mixed-integer linear optimization problem (1). The purpose of this second set of extra constraints is to ensure that any feasible solution (β, γ, y, p) of the mixed-integer linear optimization problem (1) satisfies the property that for each pair of candidates $c < c'$ that satisfy $c \equiv c'$, there exists a $k \in \{1, \ldots, |\mathcal{N}_c|\}$ that satisfies

$$\begin{gathered}
\left|\left\{b \in \{1,\ldots,B\} : \beta_{b,\mathcal{N}_c^{|\mathcal{N}_c|}} = 1\right\}\right| = \left|\left\{b \in \{1,\ldots,B\} : \beta_{b,\mathcal{N}_{c'}^{|\mathcal{N}_c|}} = 1\right\}\right| \\
\vdots \\
\left|\left\{b \in \{1,\ldots,B\} : \beta_{b,\mathcal{N}_c^{k+1}} = 1\right\}\right| = \left|\left\{b \in \{1,\ldots,B\} : \beta_{b,\mathcal{N}_{c'}^{k+1}} = 1\right\}\right| \\
\left|\left\{b \in \{1,\ldots,B\} : \beta_{b,\mathcal{N}_c^{k}} = 1\right\}\right| < \left|\left\{b \in \{1,\ldots,B\} : \beta_{b,\mathcal{N}_{c'}^{k}} = 1\right\}\right|.
\end{gathered}$$

We accomplish this by adding the following set of extra constraints (5a)-(5c) into the mixed-integer linear optimization problem (1). In the following extra constraints, we use the shorthand notation $\mathcal{I}$ to denote to the set of sequential equivalent contests,

$$\mathcal{I} \triangleq \left\{ (c,c') \in \mathcal{C} \times \mathcal{C} : \begin{array}{l} c < c',\ c \equiv c',\ \text{and there does not exist a} \\ \text{contest } \bar{c} \in \left\{c+1, \ldots, c'-1\right\} \text{ that satisfies } c \equiv \bar{c} \end{array} \right\},$$

and we use extra binary decision variables $\lambda_{c,c'}^k \in \{0,1\}$ which are added to the mixed-integer linear optimization problem (1) for each $(c,c') \in \mathcal{I}$ and $k \in \{1,\ldots,|\mathcal{N}_c|\}$.

$$\lambda_{c,c'}^{|\mathcal{N}_c|} \le \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_{c'}^{|\mathcal{N}_c|}} - \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_c^{|\mathcal{N}_c|}} \qquad \forall (c,c') \in \mathcal{I} \tag{5a}$$

$$\lambda_{c,c'}^k \le B\lambda_{c,c'}^{k+1} + \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_{c'}^k} - \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_c^k} \qquad \forall (c,c') \in \mathcal{I},\ k \in \{1,\ldots,|\mathcal{N}_c|-1\} \tag{5b}$$

$$\lambda_{c,c'}^1 = 1 \qquad \forall (c,c') \in \mathcal{I}. \tag{5c}$$

Indeed, constraints (5a) and (5b) ensure for each $(c,c') \in \mathcal{I}$ that the equality $\lambda_{c,c'}^k = 1$ can be satisfied if and only if there exists $k' \in \{k, \ldots, |\mathcal{N}_c|\}$ that satisfies the equality $\sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_c^{k''}} = \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_{c'}^{k''}}$ for all $k'' \in \{k'+1,\ldots,|\mathcal{N}_c|\}$ as well as satisfies the strict inequality $\sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_c^{k'}} < \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_{c'}^{k'}}$. Hence, constraint (5c) ensures that there exists $k' \in \{1,\ldots,|\mathcal{N}_c|\}$ that satisfies the equality $\sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_c^{k''}} = \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_{c'}^{k''}}$ for all $k'' \in \{k'+1,\ldots,|\mathcal{N}_c|\}$ as well as satisfies the strict inequality $\sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_c^{k'}} < \sum_{b\in\mathcal{B}} \beta_{b,\mathcal{N}_{c'}^{k'}}$.

5.4 Improvement 4: Heuristic for Finding Good Cuts

As our fourth step in increasing the practical efficiency of the cutting plane method, we propose a modification to the objective function of the mixed-integer linear optimization problem (2) from §4.2.2. The purpose of this modification is to guide the mixed-integer optimization problem (2) to choosing non-identity bijections in each iteration that eliminate large numbers of feasible test decks from the mixed-integer linear optimization problem (1). We demonstrate through numerical experiments in Appendix C that the proposed modification to the objective function of the mixed-integer linear optimization problem (2) can significantly decrease the number of iterations of the cutting plane method.

To motivate our subsequent developments in §5.4, we begin by presenting a framework for analyzing the quality of the non-identity bijections $\sigma \in \Sigma$ that are added to the set $\widehat{\Sigma}$ at the end of each iteration of the cutting plane method. Indeed, we recall that the practical efficiency of the cutting plane method from §4.1 depends on the number of iterations until an optimal test deck for the optimization problem (RO-Σ) is obtained. The number of iterations of the cutting plane method, in turn, depends on whether the optimization problem (CUT) in each iteration of the cutting plane method yields a non-identity bijection $\sigma \in \Sigma$ that eliminates a large number of feasible test decks from the optimization problem (RO-$\widehat{\Sigma}$). Letting $\mathscr{F}(\widehat{\Sigma})$ denote the set of test decks that are feasible for the optimization problem (RO-$\widehat{\Sigma}$), we will henceforth say (informally) that the optimization problem (CUT) yields a high-quality non-identity bijection $\sigma \in \Sigma$ if the set of feasible test decks in the next iteration $\mathscr{F}(\widehat{\Sigma} \cup \{\sigma\})$ is much smaller than the set of feasible test decks in the current iteration $\mathscr{F}(\widehat{\Sigma})$.

A priori, it might appear difficult to determine whether a non-identity bijection $\sigma \in \Sigma$ will eliminate a large number of feasible test decks from the optimization problem (RO-$\widehat{\Sigma}$). Nonetheless, we demonstrate below that high-quality non-identity bijections $\sigma \in \Sigma$ always have a structural property that we refer to as *minimal*. To define this structural property, consider any given non-identity bijection $\sigma \in \Sigma$, and let $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$ denote the undirected graph that is generated by that non-identity bijection. The set of vertices of this undirected graph is defined as the subset of contests that include a candidate that is swapped by the non-identity bijection σ,

$$\mathscr{V}^\sigma \triangleq \{c \in \mathcal{C} : \text{ there exists } i \in \mathcal{N}_c \text{ that satisfies } \sigma(i) \neq i\},$$

and the set of edges of this undirected graph is defined as the pairs of contests containing candidates that are swapped by the non-identity bijection σ,

$$\mathscr{E}^\sigma \triangleq \left\{ (c, c') \in \mathcal{C} \times \mathcal{C} : \begin{array}{c} \text{there exist } i \in \mathcal{N}_c \text{ and } i' \in \mathcal{N}_{c'} \text{ that satisfy the} \\ \text{equality } \sigma(i) = i' \text{ or satisfy the equality } \sigma(i') = i \end{array} \right\}.$$

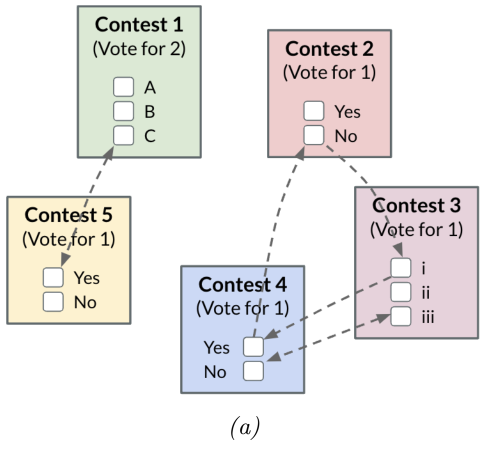
We recall from graph theory that the vertices of an undirected graph can always be partitioned into a unique collection of connected components, and we henceforth let K^σ denote the number of connected components and let $\mathscr{K}_1^\sigma, \ldots, \mathscr{K}_{K^\sigma}^\sigma \subseteq \mathscr{V}^\sigma$ denote the connected components of the undirected graph $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$.[8] Equipped with this terminology, we are ready to define the structural property of non-identity bijections that will form the basis of our subsequent discussions:

Definition 3 (Minimal)**.** We say that a non-identity bijection $\sigma \in \Sigma$ is minimal if and only if the number of connected components of $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$ satisfies $K^\sigma = 1$.

Figure 5 provides an illustration of Definition 3 by showing an example of a non-identity bijection that is not minimal. Specifically, Figure 5a presents a non-identity bijection σ in a ballot style with five contests. Figure 5b shows the undirected graph $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$ corresponding to the non-identity bijection σ. The undirected graph in Figure 5b has two connected components, which implies that the non-identity bijection from Figure 5a is not minimal.

Our main result of §5.4, which is presented below as Theorem 2, establishes the significance of non-identity bijections that are minimal. In particular, the following theorem shows that there always exists a feasible solution for the optimization problem (CUT) that is minimal. More importantly, the following Theorem 2 shows that minimal non-identity bijections are always preferred to non-minimal non-identity bijections from the perspective of eliminating the greatest number of feasible test decks from the optimization problem (RO-$\widehat{\Sigma}$).

[8]It is a straightforward exercise to show that $\mathscr{K}_1^\sigma, \ldots, \mathscr{K}_{K^\sigma}^\sigma \subseteq \mathscr{V}^\sigma$ are the connected components of the undirected graph $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$ if and only if (1) $\mathscr{K}_1^\sigma, \ldots, \mathscr{K}_{K^\sigma}^\sigma$ are disjoint, (2) the union of $\mathscr{K}_1^\sigma, \ldots, \mathscr{K}_{K^\sigma}^\sigma$ is equal to $\mathscr{V}^\sigma$, and (3) $(c, c') \in \mathscr{E}^\sigma$ implies that there exists $k \in \{1, \ldots, |K^\sigma|\}$ that satisfies $c, c' \in \mathscr{K}_k^\sigma$.



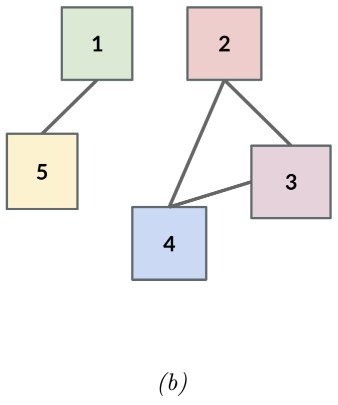


Figure 6: Example of a non-identity bijection that is not minimal in a ballot style with five contests.

Theorem 2. *Let $\sigma \in \Sigma$ denote a feasible solution for the optimization problem (CUT). For each $k \in \{1, \ldots, K^\sigma\}$, let $\sigma_k : \mathcal{N} \to \mathcal{N}$ be defined for each $c \in \mathcal{C}$ and $i \in \mathcal{N}_c$ by*

$$\sigma_k(i) \triangleq \begin{cases} \sigma(i), & \text{if } c \in \mathscr{K}_k^\sigma, \\ i, & \text{if } c \notin \mathscr{K}_k^\sigma. \end{cases}$$

Then $\sigma_1, \ldots, \sigma_{K^\sigma}$ are feasible solutions for the optimization problem (CUT) and

$$\bigcup_{k=1}^{K^\sigma} \mathscr{F}\left(\widehat{\Sigma} \cup \{\sigma_k\}\right) = \mathscr{F}\left(\widehat{\Sigma} \cup \{\sigma\}\right). \tag{6}$$

To appreciate the significance of Theorem 2, let us make several observations. First, we observe that each non-identity bijection $\sigma_k \in \Sigma$ can be interpreted as a restriction of the non-identity bijection $\sigma \in \Sigma$ that only affects the candidates from contests in the connected component $\mathscr{K}_k^\sigma$. Therefore, it follows that the number of connected components in the undirected graph $\mathscr{G}^{\sigma_k} \equiv (\mathscr{V}^{\sigma_k}, \mathscr{E}^{\sigma_k})$ corresponding to σ_k is equal to one, which implies that each of the non-identity bijections $\sigma_1, \ldots, \sigma_{K^\sigma} \in \Sigma$ is minimal. Second, we observe from line (6) that the inclusion $\mathscr{F}(\widehat{\Sigma} \cup \{\sigma_k\}) \subseteq \mathscr{F}(\widehat{\Sigma} \cup \{\sigma\})$ holds for each of the connected components $k \in \{1, \ldots, K^\sigma\}$. Hence, Theorem 2 implies that each of the non-identity bijections $\sigma_1, \ldots, \sigma_{K^\sigma}$ is preferred to σ from the perspective of eliminating feasible test decks from the optimization problem (RO-$\widehat{\Sigma}$).

Thus motivated, we now turn to the algorithmic question of how to find a minimal non-identity bijection that is feasible for the optimization problem (CUT). In the following Theorem 3, we show that such a minimal non-identity bijection can be found though making a simple modification to the objective function of the mixed-integer linear optimization problem (2).

Theorem 3. *Consider the following mixed-integer linear optimization problem:*

$$\begin{aligned} \underset{x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}}{\text{minimize}} \quad & \sum_{i,j \in \mathcal{N}: i \neq j} x_{i,j} \\ \text{subject to} \quad & (2a), (2b), (2c), (2d), (2e). \end{aligned} \tag{7}$$

Let $x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$ be an optimal solution of the mixed-integer linear optimization problem (7), and let $\sigma : \mathcal{N} \to \mathcal{N}$ be the function that satisfies the equality $\sigma(i) = j$ if and only if $x_{i,j} = 1$

for all $i, j \in \mathcal{N}$. Then σ is a minimal non-identity bijection that is feasible for the optimization problem (CUT).

We observe that the mixed-integer linear optimization problems (2) and (7) have the same decision variables and constraints. Hence, Theorem 3 shows that obtaining a minimal non-identity bijection that is feasible for the optimization problem (CUT) can be achieved by simply modifying the objective function of the mixed-integer linear optimization problem (2).

5.5 Improvement 5: Combining Noncompetitive Contests

As our fifth step in increasing the practical efficiency of the cutting plane method, we show that noncompetitive contests can be combined into one without loss of generality. By combining these contests, we demonstrate through numerical experiments in Appendix C that the number of iterations of the cutting plane method can be significantly decreased.

We begin by introducing the terminology and notation that will be used throughout §5.5. Indeed, let the original ballot style be denoted by the tuple $(\mathcal{N}, \mathcal{C}, \{\mathcal{N}_c\}_{c \in \mathcal{C}}, \{v_c\}_{c \in \mathcal{C}})$. For the original ballot style, we recall from §3.1 that a contest $c \in \mathcal{C}$ is noncompetitive if and only if the number of candidates in the contest $|\mathcal{N}_c|$ is equal to the maximum number of votes v_c. We represent the ballot style in which all of the noncompetitive contests from the original ballot style are combined into a single contest by the tuple $(\mathcal{N}, \widetilde{\mathcal{C}}, \{\widetilde{\mathcal{N}}_c\}_{c \in \widetilde{\mathcal{C}}}, \{\widetilde{v}_c\}_{c \in \widetilde{\mathcal{C}}})$, where

$$\widetilde{\mathcal{C}} \triangleq \{0\} \cup \{c \in \mathcal{C} : |\mathcal{N}_c| > v_c\};$$

$$\widetilde{\mathcal{N}}_c \triangleq \begin{cases} \mathcal{N}_c, & \text{if } c \neq 0, \\ \displaystyle\bigcup_{c' \in \mathcal{C} : |\mathcal{N}_{c'}| = v_{c'}} \mathcal{N}_{c'}, & \text{if } c = 0; \end{cases}$$

$$\widetilde{v}_c \triangleq \begin{cases} v_c, & \text{if } c \neq 0, \\ \left| \displaystyle\bigcup_{c' \in \mathcal{C} : |\mathcal{N}_{c'}| = v_{c'}} \mathcal{N}_{c'} \right|, & \text{if } c = 0. \end{cases}$$

We observe in the new ballot style that contest 0 denotes the contest that is constructed by combining all of the noncompetitive contests from the original ballot style. Finally, for each non-identity bijection $\sigma \in \Sigma$, let the output of a voting machine in the new ballot style whose mapping from candidates to targets is the bijection σ be given for each candidate $i \in \widetilde{\mathcal{N}}_c$ in each contest $c \in \widetilde{\mathcal{C}}$ by

$$\widetilde{T}_i^\sigma(\beta_1, \ldots, \beta_B) \triangleq \sum_{b=1}^{B} \mathbb{I}\left\{ \sigma(i) \in \beta_b \text{ and } \left| \left\{ \sigma(j) \in \beta_b : j \in \widetilde{\mathcal{N}}_c \right\} \right| \leq \widetilde{v}_c \right\}.$$

Equipped with the above notation, we now present the main result of §5.5. This main result, presented below as Proposition 3, establishes that the set of optimal solutions for the optimization problem (RO-Σ) for any given original ballot style will not change if all of the noncompetitive contests in the original ballot style are combined into a single contest.

Proposition 3. *Consider any original ballot style $(\mathcal{N}, \mathcal{C}, \{\mathcal{N}_c\}_{c \in \mathcal{C}}, \{v_c\}_{c \in \mathcal{C}})$, and let the ballot style in which all of the noncompetitive contests from the original ballot style are combined into a single contest be denoted by $(\mathcal{N}, \widetilde{\mathcal{C}}, \{\widetilde{\mathcal{N}}_c\}_{c \in \widetilde{\mathcal{C}}}, \{\widetilde{v}_c\}_{c \in \widetilde{\mathcal{C}}})$. Then the following equality holds for all $B \in \mathbb{N}$, $\beta_1, \ldots, \beta_B \subseteq \mathcal{N}$, $\sigma \in \Sigma \cup \{*\}$, and $i \in \mathcal{N}$:*

$$T_i^\sigma(\beta_1, \ldots, \beta_B) = \widetilde{T}_i^\sigma(\beta_1, \ldots, \beta_B).$$

We conclude §5.5 by discussing why combining the noncompetitive contests into a single contest can decrease the number of iterations of the cutting plane method. In essence, the value of combining the noncompetitive contests stems from the second improvement to the cutting plane method that is proposed in §5.2. Indeed, we recall from §5.2 that our second improvement to the cutting plane method consisted of adding the following set of extra constraints into the optimization problem (RO-$\widehat{\Sigma}$):

$$|\{b \in \{1,\dots,B\} : i \in \beta_b\}| < |\{b \in \{1,\dots,B\} : j \in \beta_b\}| \quad \forall c \in \mathcal{C}, i,j \in \mathcal{N}_c : i < j. \tag{8}$$

The constraints (8) ensure that the optimization problem (RO-$\widehat{\Sigma}$) in each iteration of the cutting plane method outputs a test deck in which candidates in the same contest receive a strictly increasing number of votes. In view of our recollection of the second improvement from §5.2, we conclude the present §5.5 with an example which shows that combining the noncompetitive contests into a single contest can decrease the number of iterations of the cutting plane method.

Example 1. Consider an original ballot style consisting of two contests, where the first contest is defined by the equalities $\mathcal{N}_1 = \{1\}$ and $v_1 = 1$, and the second contest is defined by the equalities $\mathcal{N}_2 = \{2,3\}$ and $v_2 = 2$. We observe that each of the two contests is a noncompetitive contest.

We first analyze the number of iterations of the cutting plane method in the case where the noncompetitive contests are combined into a single contest. Indeed, if the noncompetitive contests are combined into a single contest, then we observe that the new ballot style $(\mathcal{N}, \widetilde{\mathcal{C}}, \{\widetilde{\mathcal{N}}_c\}_{c\in\widetilde{\mathcal{C}}}, \{\widetilde{v}_c\}_{c\in\widetilde{\mathcal{C}}})$ consists of a single contest, $\widetilde{\mathcal{C}} = \{0\}$, wherein the candidates in that contest are given by $\widetilde{\mathcal{N}}_0 = \{1,2,3\}$ and the maximum number of votes in that contest is given by $\widetilde{v}_0 = 3$. In the first iteration of the cutting plane method, we start with $\widehat{\Sigma} = \emptyset$, in which the optimization problem (RO-$\widehat{\Sigma}$) with the constraints (8) can be written as

$$\begin{aligned} \underset{B\in\mathbb{N},\, \beta_1,\dots,\beta_B\in\mathscr{B}}{\text{minimize}} \quad & B \\ \text{subject to} \quad & T_1^*(\beta_1,\dots,\beta_B) < T_2^*(\beta_1,\dots,\beta_B) < T_3^*(\beta_1,\dots,\beta_B). \end{aligned} \tag{9}$$

We observe from inspection that the optimization problem (9) has two optimal solutions, which are stated as follows:

$$\begin{aligned} (B^1, \beta_1^1, \beta_2^1) &= (2, \{2,3\}, \{3\}), \\ (B^2, \beta_1^2, \beta_2^2) &= (2, \{3\}, \{2,3\}). \end{aligned}$$

In particular, we observe that both of those optimal solutions are feasible solutions of the optimization problem (RO-Σ). Hence, if the noncompetitive contests are combined into a single contest, then we observe for this example that the cutting plane method will terminate after a single iteration.

We conclude Example 1 by showing that the number of iterations of the cutting plane method will always be strictly greater than one if the noncompetitive contests are not combined into a single contest. Indeed, suppose that we apply the cutting plane method to the original ballot style $(\mathcal{N}, \mathcal{C}, \{\mathcal{N}_c\}_{c\in\mathcal{C}}, \{v_c\}_{c\in\mathcal{C}})$. In the first iteration of the cutting plane method, we start with $\widehat{\Sigma} = \emptyset$, in which the optimization problem (RO-$\widehat{\Sigma}$) with the constraints (8) can be written as

$$\begin{aligned} \underset{B\in\mathbb{N},\, \beta_1,\dots,\beta_B\in\mathscr{B}}{\text{minimize}} \quad & B \\ \text{subject to} \quad & T_2^*(\beta_1,\dots,\beta_B) < T_3^*(\beta_1,\dots,\beta_B). \end{aligned} \tag{10}$$

We observe from inspection that the optimization problem (10) has two optimal solutions, which are stated as follows:

$$(B^1, \beta_1^1) = (1, \{3\}),$$
$$(B^2, \beta_1^2) = (1, \{1, 3\}).$$

However, neither of those optimal solutions are feasible solutions for the optimization problem (RO-Σ).[9] Hence, if the noncompetitive contests are not combined into a single contest, then we observe for this example that the cutting plane method will always require at least two iterations. □

6 Numerical Experiments in Real-World Election

In partnership with the Michigan Bureau of Elections, we applied our approach to each of the state's 6928 ballot styles from the November 2022 general election. Through conversations with state leadership, local election officials, and vendors, we found that the ease of deploying our approach depended on two main factors: the length of the test decks with rigorous security guarantees obtained by solving our optimization problem (RO-Σ), and the computation time required by our algorithm to find optimal test decks for all 6928 ballot styles. We report below on the performance of our approach with respect to those two factors.

6.1 Length of Optimal Test Decks

The length of test decks is a crucial factor in conducting LAT in real-world elections. Long decks pose challenges, both in terms of cost and difficulty for election officials. Consequently, the practicality of our approach to achieving rigorous security guarantees in LAT depends on whether the test decks obtained by solving the optimization problem (RO-Σ) are significantly longer than the heuristic-based test decks that would otherwise be used.

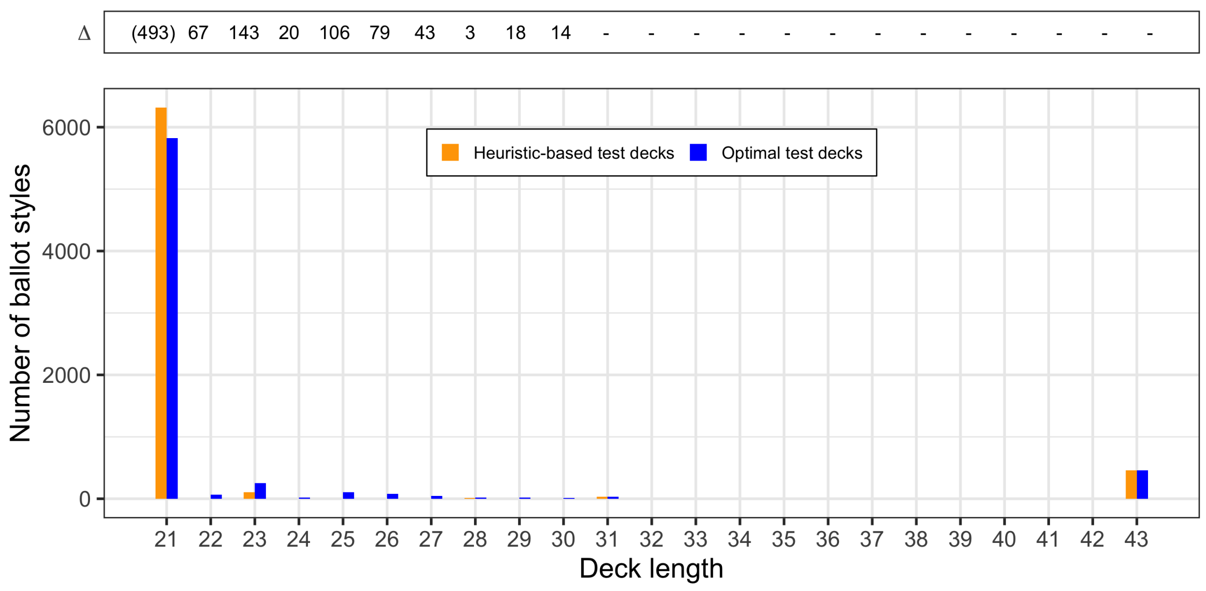
To evaluate the practicality of our approach in application to Michigan's November 2022 general election, we performed the following steps. First, we calculated the lengths of optimal test decks for each of the 6928 ballot styles by solving the optimization problem (RO-Σ) once per ballot style. To comply with minimum requirements and guidance in the state of Michigan, the following rules were added as constraints into the optimization problem (RO-Σ) (see Appendix D):

- "A different number of valid votes shall be assigned to each candidate for an office, and for and against each question" [21, MCL 168.798(1)].
- "None of the candidates, write-in positions, or proposals shall have an accumulated vote total of zero" [21, R168.773 - Rule 3(10)(a)].

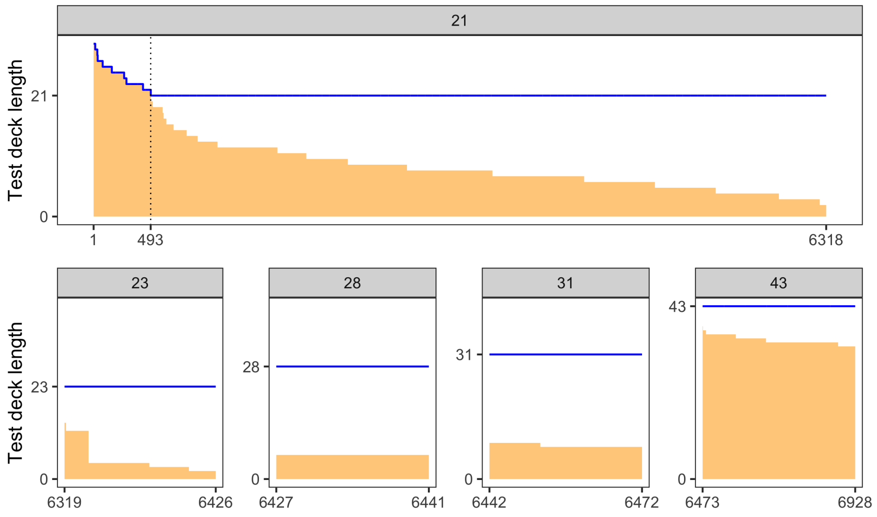
Second, we calculated the lengths of heuristic-based test decks for each of the 6928 ballot styles. Our implemented heuristic involves selecting a test deck with the minimum possible length that fulfills [21, MCL 168.798(1)] and [21, R168.773 - Rule 3(10)(a)] for each specific ballot style.[10] In comparison to the optimal test decks that were obtained by solving the optimization problem (RO-Σ), the heuristic-based test decks are not feasible solutions for (RO-Σ) and do not offer rigorous security

[9] The fact that neither $(B^1, \beta_1^1) = (1, \{3\})$ nor $(B^2, \beta_1^2) = (1, \{1, 3\})$ is a feasible solution of the optimization problem (RO-Σ) follows immediately from the fact that the optimal objective value of the optimization problem (RO-Σ) is equal to two.

[10] We compute the minimal possible length of a legally-compliant test deck for each ballot style as the maximum of $|\mathcal{N}_c|$ and $\lceil |\mathcal{N}_c|(|\mathcal{N}_c| + 1)/2v_c \rceil$ over all contests c in the ballot style, where $|\mathcal{N}_c|$ is the number of candidates in the contest and v_c is the maximum number of candidates that can be selected in the contest per ballot.



(a)



(b)

Figure 7: **(a)** *Distributions of lengths of heuristic-based test decks (orange) and optimal test decks (blue) across the 6928 ballot styles. Top of figure shows the changes in number of ballot styles (blue minus orange). The similarity of the two distributions indicates that our approach requires only minor increases in test deck length, and only for those test decks which were the shortest to begin with.* **(b)** *Length of the optimal test deck (blue) and the number of candidates in noncompetitive contests (orange) for each of the 6928 ballot styles used in Michgian's 2022 general election. The ballot styles are split across subgraphs according to the number of ballots in the heuristic-based test deck, where the top graph shows the ballot styles where the heuristic-based test decks satisfy $H^k = 21$, and bottom four graphs show the ballot styles where the heuristic-based test decks satisfy $H^k \in \{23, 28, 31, 43\}$. Results show that $O^k = \max\{H^k, NC^k\}$ for all ballot styles in this election.*

guarantees for any practically important class of cyberattacks. We note that the lengths of these heuristic-based test decks serve as lower bounds on the lengths of test decks that could be obtained by *any* heuristic that complies with [21, MCL 168.798(1)] and [21, R168.773 - Rule 3(10)(a)].

Figure 6a compares the distributions of the lengths of optimal test decks and the lengths of heuristic-based test decks across the 6928 ballot styles from Michigan's November 2022 general election. A priori, one might have anticipated that test decks that provide rigorous security guarantees would contain significantly more ballots than the shortest test decks that satisfy a state's minimum legal requirements. However, the numerical results of our experiments in Figure 6a show this is not the case. The results from Figure 6a for Michigan's November 2022 general election show that the optimal test decks obtained by solving the optimization problem (RO-Σ) require only 1.2% more ballots on average than the heuristic-based test decks across the 6928 ballot styles. Moreover, the optimal test decks require the same number of ballots as the heuristic-based test decks for all but 493 of the 6928 ballot styles. These results suggest that the rigorous security guarantees of our robust optimization approach to designing test decks can be enjoyed with essentially no additional cost or difficulty to election officials for performing LAT.

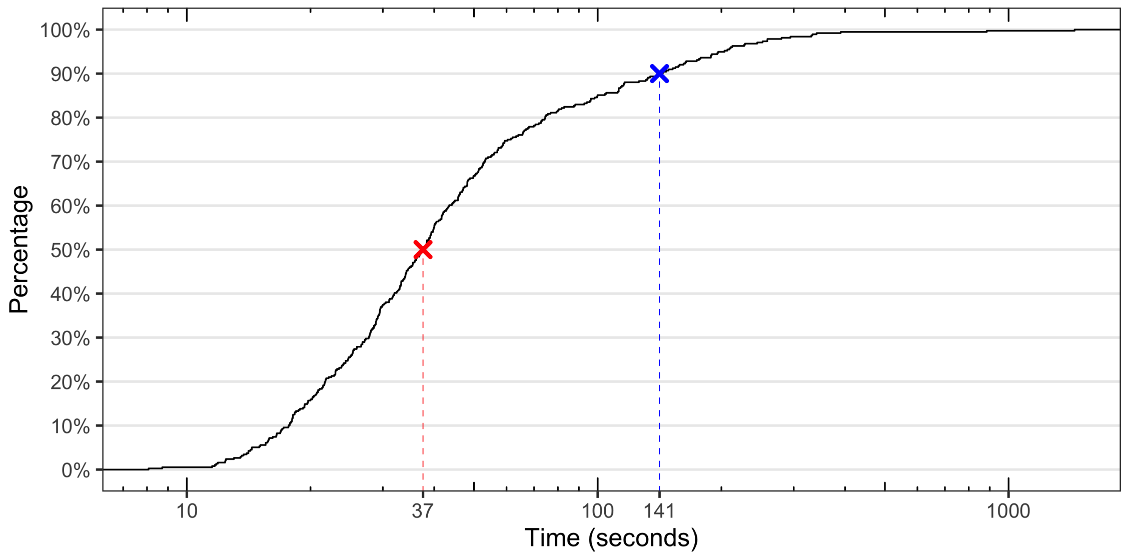
Furthermore, we find that the increases in test deck lengths in 493 of the 6928 ballot styles can be explained by a simple mathematical formula. To present this formula, let $H^1, \ldots, H^{6928} \geq 0$ denote the lengths of the heuristic-based test decks and $O^1, \ldots, O^{6928} \geq 0$ denote the lengths of the optimal test decks. Let a noncompetitive contest refer to any contest c in a ballot style in which the maximum number of candidates that a voter is allowed to select, denoted by v_c, is equal to the number of candidates in the contest, denoted by $|\mathcal{N}_c|$. In order for a test deck to satisfy the minimum legal requirement [21, MCL 168.798(1)], we observe the test deck must assign a different number of votes to each candidate within each noncompetitive contest. Moreover, we prove in §5.5 that any feasible solution of the optimization problem (RO-Σ) must assign a different number of votes to each candidate across *all* noncompetitive contests. Letting $NC^1, \ldots, NC^{6928}$ denote the number of candidates in noncompetitive contests, where $NC^k = \sum_{c:|\mathcal{N}_c|=v_c} |\mathcal{N}_c|$ for each ballot style k, we show in Figure 6b that the formula $O^k = \max\left\{H^k, NC^k\right\}$ is satisfied for all ballot styles $k = 1, \ldots, 6928$. In other words, Figure 6b shows that the optimization problem (RO-Σ) yielded test decks of an equal length to current practice for every ballot style, except for the 493 ballot styles which require longer test decks to distinguish candidates in noncompetitive contests.

Remark 5. When imposing [21, MCL 168.798(1)] and [21, R168.773 - Rule 3(10)(a)], we note that it is possible to construct ballot styles for which the formula $O = \max\{H, NC\}$ does not hold. For example, consider a ballot style comprised of two contests, each with two candidates and a maximum vote of one ($|\mathcal{N}_1| = |\mathcal{N}_2| = 2$ and $v_1 = v_2 = 1$). For this ballot style, there are $NC = 0$ candidates in noncompetitive contests, and we observe that the heuristic-based test deck requires $H = 3$ ballots. However, it follows from §5.3 that the optimal test deck for this ballot style will require at least $O \geq 4$ ballots. This example demonstrates that while the formula $O^k = \max\left\{H^k, NC^k\right\}$ explains the lengths of optimal test decks in all 6928 ballot styles from Michigan's November 2022 general election, the formula is not guaranteed to hold in general.

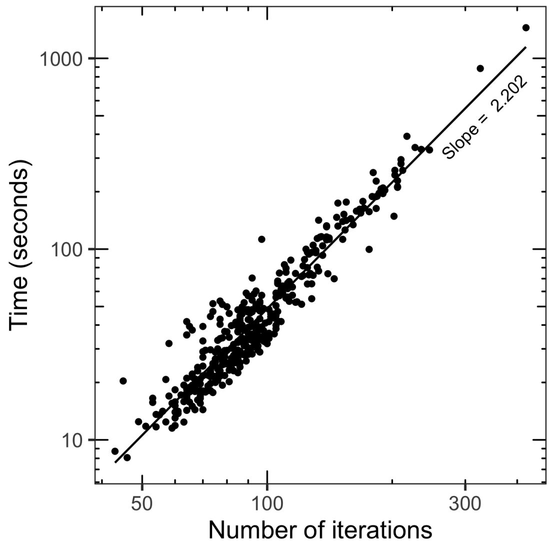
6.2 Practical Computational Time

Due to a strict schedule for finalizing ballot styles and conducting LAT, we have found that our approach must find optimal test decks for an entire state within 24-48 hours.

To apply our approach at scale, we developed strategies for reusing optimal solutions across instances of the optimization problem (RO-Σ) that corresponded to similar ballot styles, which allowed us to decrease the number of invocations of our exact algorithm from 6928 to 376 (see



(a)



(b)

Figure 8: **(a)** *Computation times for the 376 invocations of our algorithm that were run to compute optimal test decks for all 6928 ballot styles. Red indicates that 50% of the invocations required less than 37 seconds. Blue indicates that 90% of the invocations required less than 141 seconds. Total computation time was 6 hours and 42 minutes.* **(b)** *Scatterplot of the number of iterations and total computation time of our algorithm for each of the 376 invocations. Diagonal line shows the function* $\text{Time} = \exp(-6.255 + 2.202 \log(\text{Iterations}))$ *with coefficient of determination* $R^2 = 0.9426$.

Appendix E). Applying those strategies, our exact algorithm from §4 computed optimal test decks for all 6928 ballot styles in less than seven hours on a home computer. Figure 7a illustrates the distribution of computation times across the 376 invocations of our algorithm, showing that 90% of the invocations required less than 2.5 minutes. These findings demonstrate that our exact algorithm from §4 can find optimal test decks for all of the ballot styles across a state in practical computation times. Figure 7b displays the number of iterations and the total computation time required for each of the 376 invocations of our algorithm. The results of Figure 7b show that the computation time of the algorithm is driven by the number of iterations of our cutting plane method, demonstrating the value of the algorithmic developments in §5 which decrease the number of iterations of the cutting plane method.

7 Conclusion

This paper describes the first formal procedure for detecting cyberattacks in computerized voting machines *prior* to their use in elections. We achieve this by applying rigorous scientific reasoning to a widely used pre-election procedure, Logic and Accuracy Testing, which for more than a century has been performed using human intuition and simple heuristics. Unlike the longstanding practice of LAT, our approach provides a guarantee that LAT will detect any misconfiguration that swaps voting targets between candidates, whether those misconfigurations were induced deliberately or by human error. Such misconfigurations have occurred accidentally in recent elections in Michigan, Pennsylvania, and Georgia. Although these errors were later caught and corrected, they generated negative publicity, hurt public confidence, and served as the basis for a draft executive order which would have instructed the military to seize voting machines nationwide. We showed in §2 that similar misconfigurations could be strategically induced by technically unsophisticated adversaries to undermine public trust or change the outcome of an election. By applying tools from robust optimization to LAT, this paper offers a practical and scientifically rigorous way to defend against the aforementioned risks in future elections, and demonstrates that advanced computational tools can be used to realize novel benefits to public institutions.

Through our partnership with the Michigan Bureau of Elections, we found that our approach offered valuable security guarantees with only a 1.2% average increase in the number of test ballots compared to existing testing procedure in the state's November 2022 general election. Coupled with the practical computation time of our algorithm, we conclude that our approach to obtaining rigorous security guarantees with LAT is well suited to deployment throughout the United States. We hope that other states and countries will adopt our approach as a low-cost tool to improving the security and increasing public confidence in election outcomes.

There are many interesting directions for future work.

- First, we foresee ways that randomization can be used to generate short test decks with probabilistic security guarantees. We provide evidence in §6 that for states with legal requirements like Michigan's, the benefit of using randomization to design short test decks would be minuscule. This is because the test decks produced by our deterministic approach in Michigan's November 2022 election were only 1.2% longer on average than the shortest test decks that satisfy Michigan's minimal legal requirements. However, randomization may be useful for designing short test decks when considering more expansive classes of uncertainty sets (i.e., uncertainty sets that go beyond incorrect bijection mappings). Randomization could also help facilitate adoption of the robust optimization approach in states that currently have weak legal requirements that must be satisfied by test decks (e.g., states that do not require every candidate within a contest to have a distinct number of votes), as election officials in such states would be accustomed to shorter test decks.

- It is straightforward to see that the length of test decks could theoretically decrease if an election official could output the vote totals after each ballot is fed into the voting machine. However, it takes significant time for the voting machine to print the poll tape (i.e., the grocery store-like receipt that the machine prints out to show the vote totals) and significant time for the election official to then reset the machine after it prints a poll tape in order for the machine to scan more ballots. From conversations with election officials, we learned that a test deck that requires more than a few poll tapes to be printed is viewed by election officials as too time consuming to perform, too different from the current practice of logic and accuracy testing, and would thus be unlikely to be followed by election officials. In view of these practical considerations, an interesting future direction would be to characterize the savings that could be obtained in test deck length if election officials were required to print the poll tape a small (but greater than one) number of times during testing.
- On the theoretical side, many open questions remain about the computational complexity of the robust optimization problem, whether it would be possible to design approximation algorithms, and if special cases of the robust optimization problem can be solved in polynomial time.

Acknowledgements

The authors are grateful to the Michigan Department of State Bureau of Elections for their partnership in this research. This material is based upon work supported by the U.S. National Science Foundation under Grant No. CNS-1518888. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] AP. Officials: Clerk error behind county results favoring Biden. *Associated Press News*, November 2020.

[2] Arizona Secretary of State. Arizona elections procedures manual, 2019. URL https://azsos.gov/sites/default/files/2019_ELECTIONS_PROCEDURES_MANUAL_APPROVED.pdf.

[3] Arkansas State Board of Election Commissioners. County board of election commissioners procedures manual, 2022. URL https://static.ark.org/eeuploads/elections/2022_CBEC_Manual_FINAL.pdf.

[4] James Barron. On the ballot: Clean air, clean water, green jobs. *New York Times*, November 2022.

[5] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*, volume 28. Princeton University Press, 2009.

[6] Matthew Bernhard, Josh Benaloh, J Alex Halderman, Ronald L Rivest, Peter YA Ryan, Philip B Stark, Vanessa Teague, Poorvi L Vora, and Dan S Wallach. Public evidence from

secret ballots. In *Electronic Voting: Second International Joint Conference*, pages 84–109. Springer, 2017.

[7] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[8] Mark Bowden and Matthew Teague. How a Michigan clerk got embroiled in Trump's attempt to overturn the election. *Time*, December 2021.

[9] Talha Khan Burki. Diesel cars and health: the Volkswagen emissions scandal. *The Lancet Respiratory Medicine*, 3(11):838–839, 2015.

[10] Nick Corasaniti. A Pennsylvania county's election day nightmare underscores voting machine concerns. *New York Times*, November 2019.

[11] PLLC DePerno Law Office. Dominion. URL https://www.depernolaw.com/dominion.html.

[12] Georgia Voter Registration & Elections - DeKalb County. Board of registration and elections meeting notes, July 2022. URL https://www.dekalbcountyga.gov/sites/default/files/users/user3597/Board%20Materials%202022-07-14.pdf.

[13] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[14] J. Alex Halderman. Analysis of the Antrim county, Michigan November 2020 election incident. In *Proceedings of the 31st USENIX Security Symposium*, pages 589–605. USENIX Association, 2022.

[15] Douglas Jones and Barbara Simons. *Broken ballots: Will your vote count?* CSLI Publications, 2012.

[16] James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[17] David C Kimball and Martha Kropf. Voting technology, ballot measures, and residual votes. *American Politics Research*, 36(4):479–509, 2008.

[18] Puneet Kollipara. On election day, science also on the ballot in some states. *Science*, November 2016.

[19] Mark Lindeman and Philip B Stark. A gentle introduction to risk-limiting audits. *IEEE Security & Privacy*, 10(5):42–49, 2012.

[20] Christine Mai-Duc. California recall puts governor's pandemic leadership to the test. *Wall Street Journal*, July 2021.

[21] *Test Procedure Manual for Tabulators and Voter Assist Terminals (VAT)*. Michigan Department of State, Bureau of Elections, 2019. URL https://www.michigan.gov/-/media/Project/Websites/sos/01vanderroest/TEST_DECK_MANUAL05.pdf.

[22] National Conference of State Legislators. Initiative and referendum processes, January 2022. URL https://www.ncsl.org/elections-and-campaigns/initiative-and-referendum-processes.

[23] John Oliver. Voting machines: Last Week Tonight with John Oliver (HBO). URL https://www.youtube.com/watch?v=svEuG_ekNT0.

[24] Michael P Olson and Andrew R Stone. The incumbency advantage in judicial elections: Evidence from partisan trial court elections in six US states. *Political Behavior*, pages 1–22, 2022.

[25] John Sakellariadis. Voting machine trouble in Pennsylvania county triggers alarm ahead of 2024. 11 2023. URL https://www.politico.com/news/2023/11/25/voting-machine-trouble-pennsylvania-00128554.

[26] Peter Slevin. The abortion fight has voters turning to ballot initiatives. *The New Yorker*, June 2023.

[27] Betsy Woodruff Swan. Read the never-issued trump order that would have seized voting machines. *Politico*, January 2022.

[28] Carter Walker. Missing voting machine documents raise concern about Pa. county's testing processes. *Votebeat*, April 2024. URL https://www.votebeat.org/pennsylvania/2024/04/18/northampton-county-election-machine-logic-accuracy-testing-lapses-2023/.

[29] Josiah Walker, Nakul Bajaj, Braden L Crimmins, and J Alex Halderman. Logic and accuracy testing: A fifty-state review. In *Electronic Voting: Seventh International Joint Conference*, volume 13553 of *Lecture Notes in Computer Science*. International Joint Conference on Electronic Voting, Springer, September 2022.

[30] Edmund Whittaker. Eddington's theory of the constants of nature. *The Mathematical Gazette*, 29(286):137–144, 1945.

Appendices

The appendices have the following organization:

- Appendix A proposes two simple heuristics for obtaining feasible solutions for the optimization problem (RO-Σ). Using real-world data, we show that these simple heuristics will result in test decks that contain too many ballots to be used in practice.
- Appendix B describes the capabilities of an adversary who chooses both the voting machine's configuration and the test deck used in LAT.
- Appendix C contains additional numerical experiments that showcase the value of the five improvements from §5 on the practical efficiency of the cutting plane method from §4.
- Appendix D shows that various state-level legal requirements on the design of test decks can be enforced either by adding constraints into the optimization problem (RO-Σ) or by augmenting the output of the optimization problem (RO-Σ).
- Appendix E identifies circumstances in which an optimal test deck for the optimization problem (RO-Σ) for one ballot style can be efficiently translated into an optimal test deck for another similar ballot style.
- Appendix F contains the proofs of the paper's technical results.

A Upper Bounds

Our exact algorithm for solving the optimization problem (RO-Σ) is found in §4. In this appendix, we motivate the exact algorithm by presenting and analyzing two simple heuristics for the optimization problem (RO-Σ). These two heuristics, which can be found below in Propositions 4 and 5, obtain a feasible solution for the optimization problem (RO-Σ) by constructing a test deck that contains a distinct positive number of votes for each candidate. Our purpose for presenting these heuristics is (a) to show that the optimization problem (RO-Σ) always has a feasible solution and (b) to show using real-world data that heuristics based on assigning a distinct number of votes for each candidate will result in test decks that contain too many ballots to be implementable in practice.

Our first simple heuristic for the optimization problem (RO-Σ) is stated formally in the proof of the following Proposition 4. The heuristic consists of constructing a test deck in which each filled-out ballot in the test deck contains a vote for exactly one candidate (i.e., $|\beta_1| = \cdots = |\beta_B| = 1$) and in which each candidate $i \in \mathcal{N}$ is selected in exactly i of the filled-out ballots. The fact that this heuristic yields a test deck that is feasible for the optimization problem (RO-Σ) is shown in the proof of Proposition 4 to follow from Corollary 1 coupled with the fact that the heuristic gives a distinct total number of votes to each of the candidates. More generally, this heuristic is useful because it yields a simple, closed-form upper bound on the length of optimal test decks for the optimization problem (RO-Σ).

Proposition 4. *There exists a feasible solution for the optimization problem* (RO-Σ) *that satisfies* $B \le N(N+1)/2$.

Our second heuristic for the optimization problem (RO-Σ) can be viewed as a refinement of the first heuristic from Proposition 4. Like the first heuristic, our second heuristic yields a test deck that is feasible for the optimization problem (RO-Σ) by giving a distinct positive number of votes to each candidate. However, our second heuristic assigns a different positive number of votes to each candidate in such a way that allows the votes to be packed into the fewest number of ballots. More specifically, our second heuristic consists of solving the following optimization problem (11) to find a minimum-length test deck that assigns a distinct positive number of votes to each of the candidates across each of the contests:

$$\begin{aligned}
&\underset{B\in\mathbb{N},\, \beta_1,\ldots,\beta_B \in \mathscr{B}}{\text{minimize}} && B && && \text{(11a)}\\
&\text{subject to} && |\{b \in \{1,\ldots,B\} : i \in \beta_b\}| \neq |\{b \in \{1,\ldots,B\} : j \in \beta_b\}| && \forall i,j \in \mathcal{N} : i \neq j && \text{(11b)}\\
& && |\{b \in \{1,\ldots,B\} : i \in \beta_b\}| \geq 1 && \forall i \in \mathcal{N}. && \text{(11c)}
\end{aligned}$$

Indeed, constraint (11b) ensures that the test deck gives a distinct number of votes to each candidate, and constraint (11c) ensures that each candidate receives at least one vote. It follows immediately from Corollary 1 that any test deck that is feasible for the optimization problem (11) is feasible for the optimization problem (RO-Σ). Hence, the optimization problem (11) provides the tightest upper bound on the optimal objective value of the optimization problem (RO-Σ) that can be obtained by a test deck that assigns a distinct positive number of votes to each of the candidates across all of the contests.

In the following Proposition 5, we show that the optimization problem (11) can be reformulated as a mixed-integer linear optimization problem. In contrast to the optimization problem (11), the mixed-integer linear optimization problem (12) from the following Proposition 5 can be easily implemented and solved using widely available open-source and commercial optimization software such as Gurobi and Mosek.

Proposition 5. *The optimal objective value of the optimization problem* (11) *is equal to the optimal objective value of the following mixed-integer linear optimization problem* (12)*, and every optimal solution for* (12) *can be transformed into an optimal solution for* (11)*.*

$$\begin{aligned}
\underset{B\in\mathbb{N},\gamma\in\{0,1\}^{\mathcal{C}\times\mathcal{N}}}{\text{minimize}} \quad & B && && \text{(12a)}\\
\text{subject to} \quad & \sum_{g\in\mathcal{N}} \gamma_{c,g} = |\mathcal{N}_c| && \forall c \in \mathcal{C} && \text{(12b)}\\
& \sum_{c\in\mathcal{C}} \gamma_{c,g} = 1 && \forall g \in \mathcal{N} && \text{(12c)}\\
& B \geq \frac{1}{v_c}\sum_{g\in\mathcal{N}} g\gamma_{c,g} && \forall c \in \mathcal{C} && \text{(12d)}\\
& B \geq N. && && \text{(12e)}
\end{aligned}$$

Let us provide an interpretation of the decision variables and constraints of the mixed-integer linear optimization problem (12). Each binary decision variable $\gamma_{c,g}$ will be equal to one if and only if there exists a candidate $i \in \mathcal{N}_c$ in contest c that satisfies the equality $|\{b \in \{1,\ldots,B\} : i \in \beta_b\}| = g$. Constraints (12b) and (12c) ensure that a distinct number of votes are given to each of the candidates across each of the contests. Constraint (12d) enforces, for each contest $c \in \mathcal{C}$, the fact that a test deck $\beta_1,\ldots,\beta_B \in \mathscr{B}$ needs to be comprised of at least $B \geq \lceil \frac{1}{v_c}\sum_{g\in\mathcal{N}} g\gamma_{c,g} \rceil$ ballots in order for it to be possible for the test deck to give $\sum_{g\in\mathcal{N}} g\gamma_{c,g}$ votes to the candidates in contest c without causing an overvote for that contest in any of the ballots. Constraint (12e) enforces that we include at least the N ballots which are necessary for the candidate who receives N votes.

We conclude Appendix A by showing using real-world data that any heuristic that assigns a distinct positive number of votes to each candidate across all of the contests will result in test decks that contain too many ballots to be useful in practice. Specifically, we applied our second heuristic to the 6928 ballot styles that appeared in the state of Michigan in the November 2022 general election. In Figure 9, we compare the number of ballots for test decks that are optimal for the optimization problem (RO-Σ) and the number of ballots for test decks that are optimal for the optimization problem (12). The results of Figure 9 thus demonstrate that test decks that assign distinct positive numbers of votes for candidates across contests can require significantly (2.46x to 6.38x) more ballots than the test decks obtained by solving the optimization problem (RO-Σ).

B Deliberately Flawed Test Decks

In §2, we discussed a category of threats (dubbed 'Deliberately Flawed Test Decks') in which an adversary may have the opportunity to both configure the voting machine as well as design the test deck used in LAT. If this is the case, then the adversary could choose a mapping $\sigma \in \Sigma$ to use on the machine, then choose a test deck $(\beta_1,\ldots,\beta_B)$ that satisfies $T^\sigma(\beta_1,\ldots,\beta_B) = T^*(\beta_1,\ldots,\beta_B)$, thereby causing their misconfiguration to go undetected by LAT.

Such an adversary has significant freedom with respect to the $\sigma \in \Sigma$ they choose if the test deck $(\beta_1,\ldots,\beta_B)$ is not well constrained by the state's minimum legal requirements. For example, if the only minimum legal requirement on the test deck is that each candidate receives at least one vote—as is the case in a number of states [29]—the adversary can find a suitable deck for any mapping $\sigma \in \Sigma$. We formalize this observation through the following Proposition 6.

Proposition 6. *For every $\sigma \in \Sigma$, there exists a test deck $\beta_1,\ldots,\beta_B \in \mathscr{B}$ that satisfies*

$$T_i^*(\beta_1,\ldots,\beta_B) \geq 1 \;\forall i \in \mathcal{N} \text{ and } T^\sigma(\beta_1,\ldots,\beta_B) = T^*(\beta_1,\ldots,\beta_B).$$

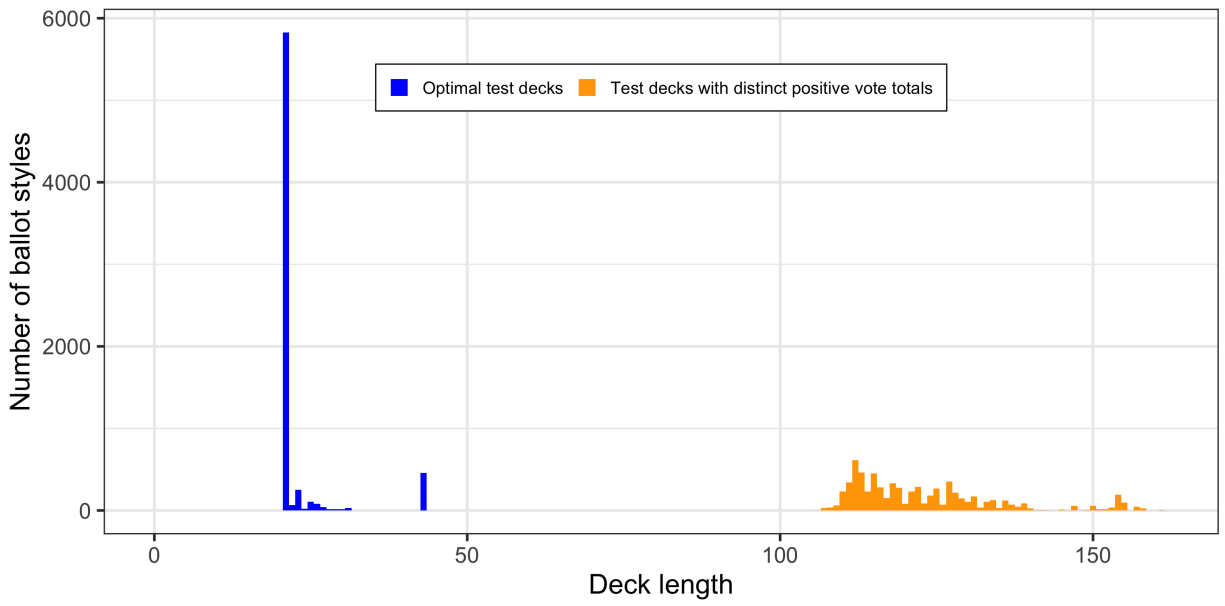


Figure 9: Distributions of lengths of test decks obtained by solving the optimization problem (RO-Σ) *(blue) and test decks obtained by solving the optimization problem* (12) *(orange) across the 6928 ballot styles. The large gap between the two distributions indicates that test decks that are optimal for the optimization problem* (RO-Σ) *require significantly fewer ballots than test decks that give a distinct positive number of votes to each candidate across all contests.*

Seventeen states also require that each candidate receives at least one vote and that no two candidates in the same contest receive the same number of votes [29]. This minimum legal requirement rules out all $\sigma \in \Sigma$ that have a cycle of swaps that includes two candidates from the same contest. For any other $\sigma \in \Sigma$, however, a suitable test deck can be generated by the adversary to hide their chosen non-identity bijection, as shown by the following Theorem 4. In the following Theorem 4 and throughout the paper, we use the notation $\sigma^n(\cdot)$ to denote the n-fold composition of $\sigma(\cdot)$.[11]

Theorem 4. *Consider any $\sigma \in \Sigma$ that satisfies the following property for all contests $c \in \mathcal{C}$, candidates $i \in \mathcal{N}_c$, and integers $n \in \mathbb{N}$:*

$$\sigma^n(i) \in \mathcal{N}_c \iff \sigma^n(i) = i.$$

Then there exists a test deck $\beta_1, \ldots, \beta_B \in \mathscr{B}$ that satisfies

$$\begin{aligned} T_i^*(\beta_1, \ldots, \beta_B) &\geq 1 \quad \forall i \in \mathcal{N} \\ T_i^*(\beta_1, \ldots, \beta_B) &\neq T_j^*(\beta_1, \ldots, \beta_B) \quad \forall c \in \mathcal{C}, i, j \in \mathcal{N}_c : i \neq j \\ T^\sigma(\beta_1, \ldots, \beta_B) &= T^*(\beta_1, \ldots, \beta_B). \end{aligned}$$

C Additional Numerical Experiments

In this appendix, we present additional numerical experiments to explore and showcase the value of the proposed improvements from §5 on the practical efficiency of the cutting plane method from §4.

[11]For example, $\sigma^1(\cdot) \triangleq \sigma(\cdot)$ and $\sigma^2(\cdot) \triangleq \sigma(\sigma(\cdot))$.

C.1 Experiment Setup

We investigate the value of the five improvements from §5 using three classes of numerical experiments. The three classes of numerical experiments are described as follows.

- *Experiment 1*: In this experiment, we generate ballot styles with varying numbers of candidates per contest. Specifically, for each $C \in \{2, \dots, 12\}$, we generate a ballot style with C contests where the contests are comprised as

$$\begin{array}{rrcrr} |\mathcal{N}_1| = 1, & |\mathcal{N}_2| = 2, & \cdots & |\mathcal{N}_{C-1}| = C-1, & |\mathcal{N}_C| = C, \\ v_1 = 1, & v_2 = 1, & \cdots & v_{C-1} = 1, & v_C = 1. \end{array}$$

- *Experiment 2*: In this experiment, we generate ballot styles with contests that have the same numbers of candidates. Specifically, for each $C \in \{2, \dots, 12\}$, we generate a ballot style with C contests where the contests are comprised as

$$\begin{array}{rrcrr} |\mathcal{N}_1| = 2, & |\mathcal{N}_2| = 2, & \cdots & |\mathcal{N}_{C-1}| = 2, & |\mathcal{N}_C| = 2, \\ v_1 = 1, & v_2 = 1, & \cdots & v_{C-1} = 1, & v_C = 1. \end{array}$$

- *Experiment 3*: In this experiment, we generate ballot styles with noncompetitive contests. Specifically, for each $C \in \{2, \dots, 12\}$, we generate a ballot style with C contests where the contests are comprised as

$$\begin{array}{rrcrr} |\mathcal{N}_1| = 1, & |\mathcal{N}_2| = 2, & \cdots & |\mathcal{N}_{C-1}| = C-1, & |\mathcal{N}_C| = C, \\ v_1 = 1, & v_2 = 2, & \cdots & v_{C-1} = C-1, & v_C = C. \end{array}$$

Our goal in each experiment is to examine the individual impact of each of the proposed improvements from §5 on the practical efficiency of the cutting plane method. To this end, we report on the performance of the following solution methods:

- *All Improvements*: In this solution method, we find an optimal solution for the optimization problem (RO-Σ) by using the cutting plane method from §4.1. In each iteration of the cutting plane method, we solve the optimization problems (RO-$\widehat{\Sigma}$) and (CUT) using the mixed-integer linear optimization reformulations (1) and (2) from §4.2. Moreover, we use each of the five improvements from §5.

- *No Improvement 1*: Same solution method as All Improvements, but we do not use Improvement 1 from §5.1.

- *No Improvement 2*: Same solution method as All Improvements, but we do not use Improvement 2 from §5.2.

- *No Improvement 3*: Same solution method as All Improvements, but we do not use Improvement 3 from §5.3.

- *No Improvement 4*: Same solution method as All Improvements, but we do not use Improvement 4 from §5.4.

- *No Improvement 5*: Same solution method as All Improvements, but we do not use Improvement 5 from §5.5.

For each $C \in \{2, \ldots, 12\}$ in each of the Experiments 1, 2, and 3, we used each of the solution methods to compute an optimal test deck. We recorded the total computation time and number of iterations of the cutting plane method for each solution method. If a solution method on a ballot style required a computation time that exceeded one hour (3600 seconds), then we terminated the solution method early without finding an optimal solution. All numerical experiments were conducted using the Gurobi optimization solver on a laptop with a 2.6 GHz 6-Core Intel Core i7 processor and 16 GB of RAM. In all experiments, we also impose a constraint in the optimization problem (RO-Σ) that each candidate must receive at least one vote (see Appendix D.1).

C.2 Results

The results of our numerical experiments from Appendix C.1 are presented in Figure 10. For visual clarity, we do not display the numerical results for No Improvement 3 in Experiments 1 and 3[12] and do not display the numerical results for No Improvement 5 in Experiments 1 and 2.[13] We reflect below on the key numerical findings from Figure 10:

- *No Improvement 1*: Experiments 1 and 3 show that the improvement from §5.1 significantly increases the practical efficiency of the cutting plane method by decreasing the per-iteration computation cost. This is seen most clearly in Experiment 3 in the ballot style with $C = 12$ contests, where the computation time of No Improvement 1 is approximately 25x greater than the computation time of All Improvements, despite both solution methods requiring only a single iteration.

- *No Improvement 2*: Experiments 1, 2, and 3 show that the improvement from §5.2 significantly increases the practical efficiency of the cutting plane method by decreasing the number of iterations. For example, in Experiment 1 in the ballot style with $C = 12$ contests, the number of iterations of No Improvement 2 is approximately 21x greater than the number of iterations of All Improvements, leading to a computation time of No Improvement 2 that is approximately 45x greater than the computation time of All Improvements. Moreover, we observe that No Improvement 2 did not terminate in less than one hour in Experiment 2 with $C \geq 9$ contests and in Experiment 3 with $C \geq 11$ contests.

- *No Improvement 3*: Experiment 2 shows that the improvement from §5.3 significantly increases the practical efficiency of the cutting plane method by decreasing the number of iterations. Indeed, in Experiment 2 in the ballot style with $C = 8$ contests, the number of iterations of No Improvement 3 is approximately 16x greater than the number of iterations of All Improvements, leading to a computation time of No Improvement 3 that is approximately 1106x greater than the computation time of All Improvements. Moreover, we observe that No Improvement 3 did not terminate in less than one hour in Experiment 2 with $C \geq 9$ contests.

- *No Improvement 4*: Experiment 2 shows that the improvement from §5.4 significantly increases the practical efficiency of the cutting plane method by decreasing the number of iterations. This is seen most clearly in Experiment 2 in the ballot style with $C = 12$ contests, where the number of iterations of No Improvement 4 is approximately 6x greater than the number of iterations of All Improvement, leading to a computation time of No Improvement

[12] Experiments 1 and 3 do not include ballot styles in which there are contests that are equivalent (see Definition 1 in §5.3), and so No Improvement 3 is identical to All Improvements in the context of Experiments 1 and 3.

[13] Experiments 1 and 2 do not include multiple noncompetitive contests (see §5.5), and so No Improvement 5 is identical to All Improvements in the context of Experiments 1 and 2.

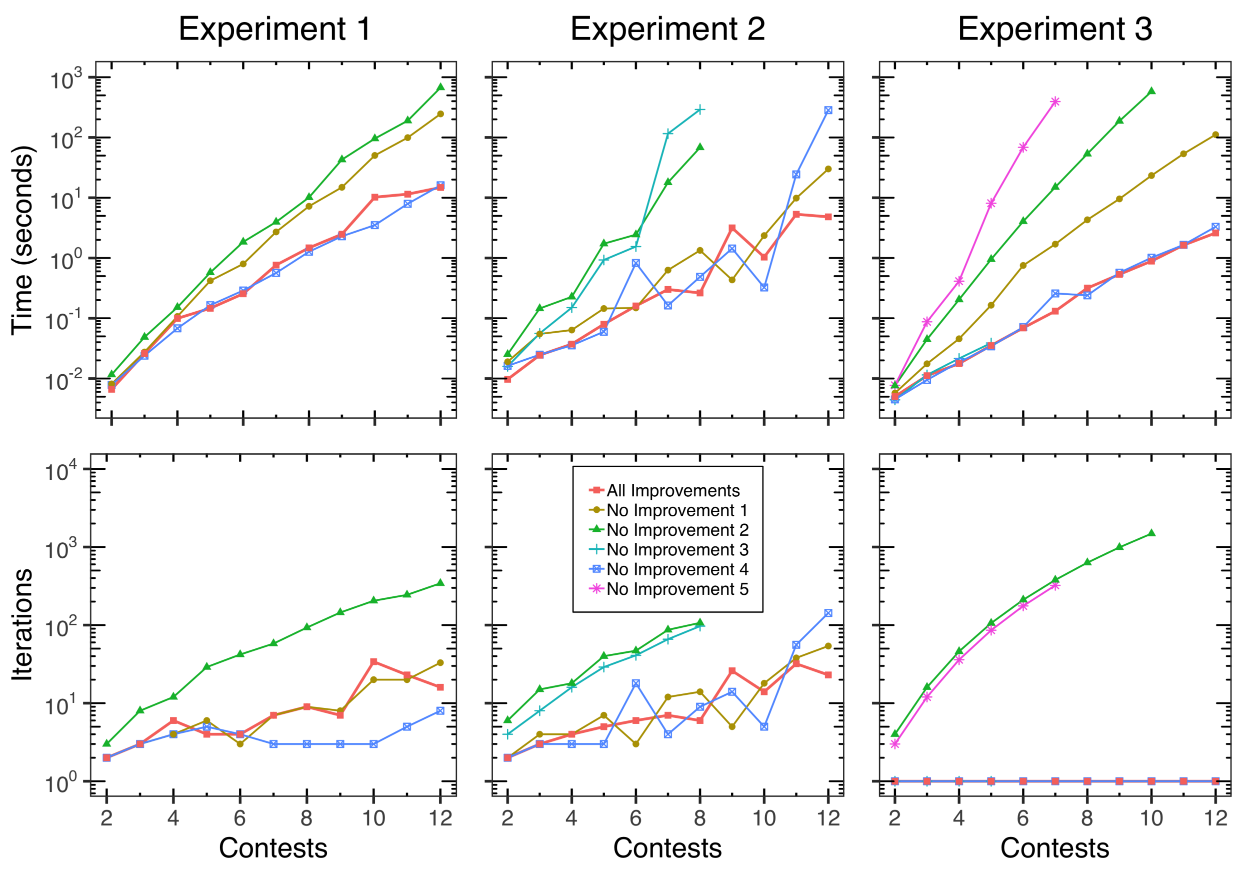


Figure 10: Numerical results for Appendix C.2.

4 that is approximately 59x greater than the computation time of All Improvements. Experiments 1 and 3, in contrast, did do show any meaningful advantages or disadvantages of using the improvement from §5.4.

- *No Improvement 5*: Experiment 3 shows that the improvement from §5.5 significantly increases the practical efficiency of the cutting plane method by decreasing the number of iterations. This is seen most clearly in Experiment 3 in the ballot style with $C = 7$ contests, where the number of iterations of No Improvement 5 is 323x greater than the number of iterations of All Improvement, leading to a computation time of No Improvement 5 that is approximately 3129x greater than the computation time of All Improvements. Moreover, we observe that No Improvement 5 did not terminate in less than one hour in Experiment 3 with $C \geq 11$ contests.

D State-Level Requirements

Each of the fifty states has minimal requirements on the design of test decks that can be legally used in LAT. For example, at least forty states have a minimum requirement that test decks must include at least one vote for each candidate on the ballot [29]. In this appendix, we demonstrate how various state-level requirements can be enforced in the optimization problem (RO-Σ).

D.1 At Least One Vote Per Candidate

At least forty states recommend that none of the candidates, write-in positions, or proposals shall have an accumulated vote total of zero [29]. To add this recommendation as a constraint into the optimization problem (RO-Σ), we add it as a constraint into the optimization problem (RO-$\widehat{\Sigma}$) in each iteration of the cutting plane method that is described in §4.1. In particular, we recall from §4.2.1 that the optimization problem (RO-$\widehat{\Sigma}$) is equivalent to the mixed-integer linear optimization problem (1), where the mixed-integer linear optimization problem (1) includes the following constraints:

$$\sum_{g \in \mathcal{B}_0} \gamma_{i,g} = 1 \qquad \forall i \in \mathcal{N} \tag{1c}$$

$$\sum_{b \in \mathcal{B}} \beta_{b,i} = \sum_{g \in \mathcal{B}_0} g\gamma_{i,g} \qquad \forall i \in \mathcal{N}. \tag{1d}$$

We recall from the discussion in §4.2.1 that the above constraints (1c) and (1d) enforce for each candidate $i \in \mathcal{N}$ that the binary decision variable $\gamma_{i,g} \in \{0,1\}$ in the mixed-integer linear optimization problem (1) will be equal to one if and only if that candidate receives exactly $g \in \mathcal{B}_0 \equiv \{0,\dots,B\}$ votes in the test deck. Therefore, to enforce that each candidate receives at least one vote in the test deck, we can add the following constraint to the mixed-integer linear optimization problem (1):

$$\gamma_{i,0} = 0 \quad \forall i \in \mathcal{N}. \tag{13}$$

D.2 Distinct Votes for Candidates in the Same Contest

At least seventeen states (including Michigan, see [21, MCL 168.798(1)]) have a requirement that test decks must assign a distinct number of votes to candidates in the same contest [29]. Following identical reasoning as in Appendix D.1, we observe that enforcing this requirement on test decks can be accomplished by adding constraints into the mixed-integer linear optimization problem (1) from §4.2.1. However, we recall that §5.2 offers an improvement to the cutting plane method that consists of adding the extra constraints (4) to the mixed-integer linear optimization problem (1). Because those extra constraints enforce that candidates within the same contest receive different numbers of votes, we conclude that the state requirement is accomplished by using the improvement from §5.2.

D.3 Overvoted Ballots

Several states (including Michigan, see [21, MCL 168.776 Rule 6(4)(f)]) require that the test deck include at least one ballot that contains an overvote in one or more contests [29]. As we show in the following Proposition 7, this requirement can be satisfied by solving the optimization problem (RO-Σ) to obtain a test deck, and then appending that test deck with an additional filled-out ballot that includes a vote for every candidate in every contest.

Proposition 7. *Let $\bar{\beta} \triangleq \mathcal{N}$ denote the filled-out ballot that includes a vote for every candidate in every contest. If $(B, \beta_1, \dots, \beta_B)$ is a feasible solution for the optimization problem* (RO-Σ)*, then the following holds:*

$$T^{\sigma}(\beta_1, \dots, \beta_B, \bar{\beta}) \neq T^{*}(\beta_1, \dots, \beta_B, \bar{\beta}) \quad \forall \sigma \in \Sigma.$$

In particular, the above proposition shows that if we solve the optimization problem (RO-Σ), and we then augment the optimal test deck by adding a filled-out ballot that votes for every candidate in every contest, then the augmented test deck will retain the desired security guarantee that the output $T^{\sigma}(\beta_1, \ldots, \beta_B, \bar{\beta})$ of a voting machine with any mapping $\sigma \in \Sigma$ will be different from the output $T^{*}(\beta_1, \ldots, \beta_B, \bar{\beta})$ of the voting machine that operates correctly.

While the above strategy can be used in many states including Michigan, certain states impose an additional requirement that the overvoted ballot in the test deck must cast precisely $v_c + 1$ votes in each contest $c \in \mathcal{C}$ that satisfies $v_c > |\mathcal{N}_c|$ [29]. In those states, the strategy from Proposition 7 cannot be applied when there exist contests that satisfy the inequality $v_c > |\mathcal{N}_c| + 1$, as the strategy from Proposition 7 would require the overvoted ballot to vote for strictly greater than $v_c + 1$ candidates in some contests.[14]

To develop test decks for states with the aforementioned additional requirement on the overvoted ballot, we consider the following assumption on their voting machines.

Assumption 1. When a voting machine interprets a ballot as containing an overvote in at least one contest, it will produce an "overvote alert" notification. This alert will not specify which contest(s) are interpreted as containing an overvote, but will allow for a determination of which *ballots* contain some overvoted contest.

Most modern voting machines are designed to satisfy this assumption. Indeed, in many jurisdictions, the purpose of including overvoted ballots in the test deck is precisely to evaluate whether this functionality works as expected (see, e.g., [2, p.91] or [3, p.62]). If this assumption is believed to hold, then we show in the following Proposition 8 that we can satisfy the aforementioned stricter requirement by solving the optimization problem (RO-Σ) to obtain a test deck, and then appending that test deck with an additional filled-out ballot that casts precisely $v_c + 1$ votes for each contest $c \in \mathcal{C}$ that satisfies $v_c > |\mathcal{N}_c|$.

Proposition 8. *Let $\tilde{\beta} \subseteq \mathcal{N}$ denote a filled-out ballot that satisfies the following equality for each contest $c \in \mathcal{C}$:*

$$|\tilde{\beta} \cap \mathcal{N}_c| = \begin{cases} v_c + 1, & \text{if } |\mathcal{N}_c| > v_c, \\ 0, & \text{otherwise.} \end{cases}$$

If $(B, \beta_1, \ldots, \beta_B)$ is a feasible solution for the optimization problem (RO-Σ), then for all $\sigma \in \Sigma$, at least one of the following two conditions hold:

- *There exists a contest in at least one of the filled-out ballots $\beta_1, \ldots, \beta_B \in \mathscr{B}$ that is interpreted by a voting machine with mapping σ as containing an overvote; that is, there exist $b \in \mathcal{B}$ and $c \in \mathcal{C}$ that satisfy $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| > v_c$.*
- *$T^{\sigma}(\beta_1, \ldots, \beta_B, \tilde{\beta}) \neq T^{*}(\beta_1, \ldots, \beta_B, \tilde{\beta})$.*

Proposition 8 tells us that each incorrect mapping $\sigma \in \Sigma$ will be detected by the modified deck $(\beta_1, \ldots, \beta_B, \tilde{\beta})$ if Assumption 1 holds. This is because either the output of the voting machine will differ from what is expected (i.e., $T^{\sigma}(\beta_1, \ldots, \beta_B, \tilde{\beta}) \neq T^{*}(\beta_1, \ldots, \beta_B, \tilde{\beta})$) or because some feasible ballot will produce an alert indicating that it has been overvoted (i.e., there will exist some $b \in \mathcal{B}$ and $c \in \mathcal{C}$ which satisfy $\sum_{i \in \mathcal{N}_c} \beta_{b,\sigma(i)} \geq v_c + 1$). In either event, the behavior of the machine under mapping $\sigma \in \Sigma$ will be distinguishable from the behavior of a machine which is operating correctly. This allows us to satisfy the aforementioned state requirement by including ballot $\tilde{\beta}$ while maintaining the desired security guarantees.

[14] An example of a contest that typically satisfies $v_c > |\mathcal{N}_c| + 1$ is the Presidential contest; even though voters may select up to one candidate in the presidential contest, voters will typically be allowed to select from candidates from four or more political parties.

D.4 Party-Line Option

Several states including Michigan (see [21, p.18]) provide an option to voters in certain ballot styles to choose a so-called "party-line option". A party-line option is a special target on a ballot that, if marked, defaults the ballot to selecting a specific party's candidates (e.g., Republican candidates, Democrat candidates) in each of the contests. The voting machine functionality related to the processing of party-line voting is complex due to the fact that voters can select a party-line vote but also can, if desired, override the default party selection in one or more contests. Because of this complexity, states such as Michigan provide a separate set of test deck requirements for evaluating the functionality of party-line options. Because the requirements of evaluating the party-line option are distinct from the requirements of test decks, a separate test deck than that obtained from solving the optimization problem (RO-Σ) can be constructed for testing party-option functionality.

E Solution Reuse and Translation

In §6, we discussed using our approach to generate test decks for Michigan's November 2022 general election. In this election, the state of Michigan used 6928 ballot styles. In principle, an optimal test deck for each ballot style could be found by solving the optimization problem (RO-Σ) for the specific ballot style. Solving the optimization problem (RO-Σ) once for each of the 6928 ballot styles, however, would be computationally time consuming.

In this appendix, we describe two strategies (which we refer to as 'solution reuse' and 'solution translation') that significantly reduced the computation time that was required for obtaining optimal test decks for all 6928 ballot styles. The two strategies are based on showing that an optimal solution to the optimization problem (RO-Σ) for one ballot style will, under certain conditions, be a feasible (and sometimes optimal) solution for the optimization problem (RO-Σ) for another similar ballot style. By using our two strategies to reuse and translate optimal solutions between similar ballot styles, we were able to obtain optimal test decks for all 6928 ballot styles despite solving the optimization problem (RO-Σ) to completion only 376 times.

E.1 Solution Reuse

Our first strategy consists of identifying conditions under which the optimal solution of the optimization problem (RO-Σ) for one ballot style is guaranteed to be an optimal solution for another similar ballot style. To apply this strategy, we first convert each ballot style into what we henceforth refer to as its *normal form*. Converting a ballot style into its normal form entails performing the following two transformations:

1. Combine the ballot style's noncompetitive contests into a single contest as described in §5.5.

2. Sort the indices of the contests such that $c < c'$ if $[|\mathcal{N}_c| > |\mathcal{N}_{c'}|]$ or $[|\mathcal{N}_c| = |\mathcal{N}_{c'}|$ and $v_c > v_{c'}]$.

Any optimal solution to the optimization problem (RO-Σ) for the normalized version of a ballot style can be efficiently transformed into an optimal solution for the original style, simply by reversing the translation of candidate indicies on the output β variables and separating the combined contest into its constituent components as previously described. This means we can eliminate repeated normalized forms, reducing the number of styles from 6928 to 1812.

E.2 Solution Translation

Our second strategy consists of identifying conditions under which an optimal solution for the optimization problem (RO-Σ) for one ballot style is guaranteed to be a feasible (but possibly suboptimal) solution to the optimization problem (RO-Σ) for another similar ballot style. Lemma 4 specifies the strategy in greater detail. To make the greatest use of Lemma 4, imagine that each noncompetitive contest has been split so that each candidate has a contest of their own; we can do this without loss of generality as a corollary of Proposition 3.

Lemma 4. *Consider a ballot style parameterized by* $(\mathcal{N}, \mathcal{C}, \{\mathcal{N}_c\}_{c\in\mathcal{C}}, \{v_c\}_{c\in\mathcal{C}})$*, let* $\bar{\mathcal{C}} \subset \mathcal{C}$ *denote a subset of contests, and let* $\bar{\mathcal{N}} \triangleq \bigcup_{c\in\bar{\mathcal{C}}} \mathcal{N}_c$ *denote the set of candidates in those contests. If* $(B, \beta_1, \ldots, \beta_B)$ *is an optimal solution for the optimization problem* (RO-Σ) *for the ballot style parameterized by* $(\mathcal{N}, \mathcal{C}, \{\mathcal{N}_c\}_{c\in\mathcal{C}}, \{v_c\}_{c\in\mathcal{C}})$*, then* $(B, \beta_1\backslash\bar{\mathcal{N}}, \ldots, \beta_B\backslash\bar{\mathcal{N}})$ *is a feasible solution for the optimization problem* (RO-Σ) *for the ballot style parameterized by* $(\mathcal{N}\backslash\bar{\mathcal{N}}, \mathcal{C}\backslash\bar{\mathcal{C}}, \{\mathcal{N}_c\}_{c\in\mathcal{C}\backslash\bar{\mathcal{C}}}, \{v_c\}_{c\in\mathcal{C}\backslash\bar{\mathcal{C}}})$*.*

In words, the above lemma shows that an optimal solution from a 'complex' ballot style is guaranteed to be a feasible solution for a 'simple' ballot style if (a) the competitive contests in the simpler ballot style are a subset of those in the more complex ballot style and (b) there are at least as many candidates in noncompetitive contests in the complex ballot style as in the simpler ballot style.

To understand the practical significance of Lemma 4, we recall that each iteration of the cutting plane method from §4.1 involves solving the optimization problem (RO-$\widehat{\Sigma}$) to obtain a lower bound on the optimal objective value of the optimization problem (RO-Σ). If this lower bound is ever equal to the length of some feasible solution to (RO-Σ) derived for a more complicated style, we can halt the cutting plane method early and translate a solution according to this lemma.

To utilize this second strategy in §6, we solved the optimization problem (RO-Σ) for ballot styles in decreasing order by their number of competitive contests. When two ballot styles had the same number of competitive contests, we solved the one with more candidates in noncompetitive contests first. When using our cutting plane method for each ballot style, we first identified the shortest feasible solution which is suitable for translation (if any such solution exists) from the ballot styles that were solved previously. Finally, we halted the cutting plane method early if the lower bound reached the length of that solution.

In practice, this second strategy allowed for early termination of the cutting plane method in a majority of ballot styles. Of the 6928 ballot styles and 1812 distinct normalized forms, we were able to terminate computation early in all but 376 cases. This yielded significant time savings; the average time to generate a test deck for a ballot style which terminates early is on the order of one-tenth of a second, while the average time to generate a test deck for the other 376 styles in on the order of a minute.

F Proofs

F.1 Proofs from §3.4

Proof of Theorem 1. Let $B \in \mathbb{N}$, $\beta_1, \dots, \beta_B \in \mathscr{B}$, and $\sigma \in \Sigma$.

To show the first direction of Theorem 1, suppose that the equality $|\{b \in \{1, \dots, B\} : i \in \beta_b\}| = |\{b \in \{1, \dots, B\} : \sigma(i) \in \beta_b\}|$ holds for all candidates $i \in \mathcal{N}$ and that the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c$ holds for all contests $c \in \mathcal{C}$ and all ballots $b \in \{1, \dots, B\}$. In this case, we observe for each contest $c \in \mathcal{C}$ and each candidate $i \in \mathcal{N}_c$ that

$$\begin{aligned}
T_i^\sigma(\beta_1, \dots, \beta_B) &= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\right\} \\
&= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b\right\} \\
&= |\{b \in \{1, \dots, B\} : \sigma(i) \in \beta_b\}| \\
&= |\{b \in \{1, \dots, B\} : i \in \beta_b\}| \\
&= T_i^*(\beta_1, \dots, \beta_B).
\end{aligned}$$

The first equality is the definition of $T_i^\sigma(\cdot)$ from §3.3. The second equality follows from the supposition that the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_{c'}\}| \leq v_{c'}$ holds for all contests $c' \in \mathcal{C}$ and all ballots $b \in \{1, \dots, B\}$. The third equality follows from algebra. The fourth equality follows from the supposition that the equality $|\{b \in \{1, \dots, B\} : i' \in \beta_b\}| = |\{b \in \{1, \dots, B\} : \sigma(i') \in \beta_b\}|$ holds for all candidates $i' \in \mathcal{N}$. The fifth equality follows from Remark 1 and from the fact that $\beta_1, \dots, \beta_B \in \mathscr{B}$. Because we have shown that the equality $T_i^\sigma(\beta_1, \dots, \beta_B) = T_i^*(\beta_1, \dots, \beta_B)$ holds for all candidates $i \in \mathcal{N}$, our proof of the first direction of Theorem 1 is complete.

To show the other direction of Theorem 1, suppose that the equality $T^*(\beta_1, \dots, \beta_B) = T^\sigma(\beta_1, \dots, \beta_B)$ holds. In this case, we observe that

$$\begin{aligned}
&\sum_{i \in \mathcal{N}} \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\right\} \\
&= \sum_{i \in \mathcal{N}} T_i^\sigma(\beta_1, \dots, \beta_B) \\
&= \sum_{i \in \mathcal{N}} T_i^*(\beta_1, \dots, \beta_B) \\
&= \sum_{i \in \mathcal{N}} \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\mathcal{N}_c \cap \beta_b| \leq v_c\right\} \\
&= \sum_{i \in \mathcal{N}} \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b\right\} \\
&= \sum_{i \in \mathcal{N}} \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b\right\}.
\end{aligned}$$

The first equality is the definition of $T_i^\sigma(\cdot)$. The second equality follows from the supposition that the equality $T^*(\beta_1, \dots, \beta_B) = T^\sigma(\beta_1, \dots, \beta_B)$ holds. The third equality is the definition of $T_i^*(\cdot)$. The fourth equality follows from the fact that the ballots satisfy $\beta_1, \dots, \beta_B \in \mathscr{B}$. The fifth equality follows from the fact that σ is a bijection. Combining the above equalities, we conclude that the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c$ must hold for all contests $c \in \mathcal{C}$ and all ballots $b \in \{1, \ldots, B\}$.

It remains for us to show that the equality $|\{b \in \{1,\ldots,B\} : i \in \beta_b\}| = |\{b \in \{1,\ldots,B\} : \sigma(i) \in \beta_b\}|$ holds for all candidates $i \in \mathcal{N}$. Indeed, we observe for each contest $c \in \mathcal{C}$ and each candidate $i \in \mathcal{N}_c$ that

$$\begin{aligned}
|\{b \in \{1,\ldots,B\} : \sigma(i) \in \beta_b\}| &= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b\right\} \\
&= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\} \\
&= T_i^{\sigma}(\beta_1,\ldots,\beta_B) \\
&= T_i^{*}(\beta_1,\ldots,\beta_B) \\
&= |\{b \in \{1,\ldots,B\} : i \in \beta_b\}|.
\end{aligned}$$

The first equality follows from algebra. The second equality follows from our prior conclusion that the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c$ holds for all ballots $b \in \{1,\ldots,B\}$. The third equality is the definition of $T_i^{\sigma}(\cdot)$. The fourth equality follows from the supposition that $T^{\sigma}(\beta_1,\ldots,\beta_B) = T^{*}(\beta_1,\ldots,\beta_B)$. The fifth equality follows from Remark 1 and from the fact that $\beta_1,\ldots,\beta_B \in \mathscr{B}$. Because we have shown that the equality $|\{b \in \{1,\ldots,B\} : i \in \beta_b\}| = |\{b \in \{1,\ldots,B\} : \sigma(i) \in \beta_b\}|$ holds for all candidates $i \in \mathcal{N}$, our proof of the other direction of Theorem 1 is complete. □

F.2 Proofs from §5.1

Proof of Lemma 1. Our proof consists of proving the contrapositive of the desired result. Indeed, consider any mapping $\sigma \in \widehat{\Sigma}$, contest $c \in \mathcal{C}$, and ballot $b \in \mathcal{B}$. Moreover, suppose that there exists a feasible solution of the mixed-integer linear optimization problem (1) that satisfies the equality $p^{\sigma}_{b,c} = 0$. In this case, we observe that

$$\begin{aligned}
v_c + 1 &\le \sum_{i \in \mathcal{N}_c} \beta_{b,\sigma(i)} \\
&= \sum_{c' \in \mathcal{C}} \left(\sum_{i \in \mathcal{N}_c : \sigma(i) \in \mathcal{N}_{c'}} \beta_{b,\sigma(i)} \right) \\
&= \sum_{c' \in \mathcal{C}} \min \left\{ \sum_{i \in \mathcal{N}_c : \sigma(i) \in \mathcal{N}_{c'}} \beta_{b,\sigma(i)}, v_{c'} \right\} \\
&\le \sum_{c' \in \mathcal{C}} \min\left\{ |\{i \in \mathcal{N}_c : \sigma(i) \in \mathcal{N}_{c'}\}|, v_{c'} \right\}.
\end{aligned}$$

Indeed, the first inequality follows from the fact that $p^{\sigma}_{b,c} = 0$ and constraint (1f). The first equality follows from algebra. The second equality follows from constraint (1b), which implies for each contest $c' \in \mathcal{C}$ that $\sum_{i \in \mathcal{N}_c : \sigma(i) \in \mathcal{N}_{c'}} \beta_{b,\sigma(i)} \le \sum_{i \in \mathcal{N}_{c'}} \beta_{b,i} \le v_{c'}$. The second inequality follows from algebra. □

F.3 Proofs from §5.2

Proof of Lemma 2. Consider any feasible solution of the optimization problem (RO-Σ), and let that feasible solution be denoted by $(B, \beta_1, \ldots, \beta_B)$. Suppose for the sake of developing a contradiction that there exists a contest $c \in \mathcal{C}$ and a pair of candidates $i, j \in \mathcal{N}_c$ such that $i \neq j$ and $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = |\{b \in \{1, \ldots, B\} : j \in \beta_b\}|$. In what follows, we will make use of a non-identity bijection $\sigma \in \Sigma$ constructed for each candidate $i' \in \mathcal{N}$ as follows:

$$\sigma\left(i'\right) \triangleq \begin{cases} i', & \text{if } i' \in \mathcal{N} \setminus \{i, j\}, \\ j, & \text{if } i' = i, \\ i, & \text{if } i' = j. \end{cases} \tag{14}$$

The remainder of the proof of Lemma 2 consists of showing that the equality $T^{\sigma}(\beta_1, \ldots, \beta_B) = T^*(\beta_1, \ldots, \beta_B)$ is satisfied. Indeed, we observe for each candidate $i' \in \mathcal{N}$ that

$$\begin{aligned} |\{b \in \{1, \ldots, B\} : \sigma(i') \in \beta_b\}| &= \begin{cases} |\{b \in \{1, \ldots, B\} : i' \in \beta_b\}|, & \text{if } i' \in \mathcal{N} \setminus \{i, j\}, \\ |\{b \in \{1, \ldots, B\} : j \in \beta_b\}|, & \text{if } i' = i, \\ |\{b \in \{1, \ldots, B\} : i \in \beta_b\}|, & \text{if } i' = j \end{cases} \\ &= |\{b \in \{1, \ldots, B\} : i' \in \beta_b\}|, \end{aligned}$$

where the first equality follows from our construction of $\sigma \in \Sigma$ on line (14) and the second equality follows from the supposition that $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = |\{b \in \{1, \ldots, B\} : j \in \beta_b\}|$. Moreover, we observe for each $b \in \{1, \ldots, B\}$ and each contest $c' \in \mathcal{C}$ that

$$\begin{aligned} &\left|\left\{\sigma\left(i'\right) \in \beta_b : i' \in \mathcal{N}_{c'}\right\}\right| \\ &= \begin{cases} \left|\{i' \in \beta_b : i' \in \mathcal{N}_c \setminus \{i, j\}\} \cup \{j\}\right|, & \text{if } c' = c,\ i \in \beta_b, \text{ and } j \notin \beta_b, \\ \left|\{i' \in \beta_b : i' \in \mathcal{N}_c \setminus \{i, j\}\} \cup \{i\}\right|, & \text{if } c' = c,\ i \notin \beta_b, \text{ and } j \in \beta_b, \\ \left|\{i' \in \beta_b : i' \in \mathcal{N}_{c'}\}\right|, & \text{otherwise} \end{cases} \\ &= \begin{cases} \left|\{i' \in \beta_b : i' \in \mathcal{N}_c \setminus \{i, j\}\}\right| + 1, & \text{if } c' = c,\ i \in \beta_b, \text{ and } j \notin \beta_b, \\ \left|\{i' \in \beta_b : i' \in \mathcal{N}_c \setminus \{i, j\}\}\right| + 1, & \text{if } c' = c,\ i \notin \beta_b, \text{ and } j \in \beta_b, \\ \left|\{i' \in \beta_b : i' \in \mathcal{N}_{c'}\}\right|, & \text{otherwise} \end{cases} \\ &= \left|\left\{i' \in \beta_b : i' \in \mathcal{N}_{c'}\right\}\right| \\ &\leq v_{c'}, \end{aligned}$$

where the first equality follows from our construction of $\sigma \in \Sigma$ on line (14), the second equality follows from algebra, the third equality follows from the fact that $i, j \in \mathcal{N}_c$, and the inequality follows from the fact that $\beta_1, \ldots, \beta_B \in \mathscr{B}$. Combining the above analysis with Theorem 1 from §3.4, we conclude that the equality $T^{\sigma}(\beta_1, \ldots, \beta_B) = T^*(\beta_1, \ldots, \beta_B)$ is satisfied.

Because we have shown that there exists a non-identity bijection $\sigma \in \Sigma$ that satisfies the equality $T^{\sigma}(\beta_1, \ldots, \beta_B) = T^*(\beta_1, \ldots, \beta_B)$, we have obtained a contradiction with the fact the that $(B, \beta_1, \ldots, \beta_B)$ is a feasible solution of the optimization problem (RO-Σ). Our proof of Lemma 2 is thus complete. $\square$

Proof of Proposition 1. Consider any optimal solution of the optimization problem (RO-Σ), and let that optimal solution be denoted by $(\beta_1, \ldots, \beta_B)$. We henceforth assume without loss of generality that $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \leq |\{b \in \{1, \ldots, B\} : j \in \beta_b\}|$ for all candidates $i < j$ that appear in the same contest. To see why this assumption is without loss of generality, suppose for the sake of argument that this assumption was not true. In that case, for each contest $c \in \mathcal{C}$, let $\pi_c : \mathcal{N}_c \to \mathcal{N}_c$ be a bijection that satisfies $|\{b \in \{1, \ldots, B\} : \pi_c(i) \in \beta_b\}| \leq |\{b \in \{1, \ldots, B\} : \pi_c(j) \in \beta_b\}|$ for each pair of candidates $i, j \in \mathcal{N}_c$ that satisfies $i < j$. Hence, by replacing the index of each candidate

$i \in \mathcal{N}_c$ with the index $\pi_c(i)$, we conclude that the assumption that $T_i^*(\beta_1,\ldots,\beta_B) \le T_j^*(\beta_1,\ldots,\beta_B)$ for all candidates $i < j$ that appear in the same contest can be made without loss of generality. Combining that assumption with Lemma 2, our proof of Proposition 1 is complete. □

F.4 Proofs from §5.3

Proof of Lemma 3. Consider any feasible solution of the optimization problem (RO-Σ), and let that feasible solution be denoted by $(B,\beta_1,\ldots,\beta_B)$. Suppose for the sake of developing a contradiction that there exist two contests $c < c'$ that satisfy $c \equiv c'$ and satisfy

$$\begin{aligned}\left|\left\{b \in \{1,\ldots,B\} : \mathcal{N}_c^{|\mathcal{N}_c|} \in \beta_b\right\}\right| &= \left|\left\{b \in \{1,\ldots,B\} : \mathcal{N}_{c'}^{|\mathcal{N}_c|} \in \beta_b\right\}\right|\\ &\vdots\\ \left|\left\{b \in \{1,\ldots,B\} : \mathcal{N}_c^{1} \in \beta_b\right\}\right| &= \left|\left\{b \in \{1,\ldots,B\} : \mathcal{N}_{c'}^{1} \in \beta_b\right\}\right|.\end{aligned}$$

In what follows, we will make use of a non-identity bijection $\sigma \in \Sigma$ that is defined for each candidate $i \in \mathcal{N}$ as follows:

$$\sigma(i) \triangleq \begin{cases}\mathcal{N}_c^k, & \text{if there exists } k \in \{1,\ldots,|\mathcal{N}_c|\} \text{ such that } i = \mathcal{N}_{c'}^k,\\ \mathcal{N}_{c'}^k, & \text{if there exists } k \in \{1,\ldots,|\mathcal{N}_c|\} \text{ such that } i = \mathcal{N}_c^k,\\ i, & \text{otherwise.}\end{cases} \tag{15}$$

We observe by construction that the bijection σ swaps the targets of candidates $\mathcal{N}_c^k$ and $\mathcal{N}_{c'}^k$ for each $k \in \{1,\ldots,|\mathcal{N}_c|\}$.

The remainder of the proof of Lemma 3 consists of showing that the equality $T^\sigma(\beta_1,\ldots,\beta_B) = T^*(\beta_1,\ldots,\beta_B)$ is satisfied. Indeed, we observe for each candidate $i \in \mathcal{N}$ that

$$\begin{aligned}&|\{b \in \{1,\ldots,B\} : \sigma(i) \in \beta_b\}|\\ &= \begin{cases}|\{b \in \{1,\ldots,B\} : \sigma(\mathcal{N}_{c'}^k) \in \beta_b\}|, & \text{if there exists } k \in \{1,\ldots,|\mathcal{N}_c|\} \text{ such that } i = \mathcal{N}_{c'}^k,\\ |\{b \in \{1,\ldots,B\} : \sigma(\mathcal{N}_{c}^k) \in \beta_b\}|, & \text{if there exists } k \in \{1,\ldots,|\mathcal{N}_c|\} \text{ such that } i = \mathcal{N}_{c}^k,\\ |\{b \in \{1,\ldots,B\} : \sigma(i) \in \beta_b\}|, & \text{otherwise}\end{cases}\\ &= \begin{cases}|\{b \in \{1,\ldots,B\} : \mathcal{N}_{c}^k \in \beta_b\}|, & \text{if there exists } k \in \{1,\ldots,|\mathcal{N}_c|\} \text{ such that } i = \mathcal{N}_{c'}^k,\\ |\{b \in \{1,\ldots,B\} : \mathcal{N}_{c'}^k \in \beta_b\}|, & \text{if there exists } k \in \{1,\ldots,|\mathcal{N}_c|\} \text{ such that } i = \mathcal{N}_{c}^k,\\ |\{b \in \{1,\ldots,B\} : i \in \beta_b\}|, & \text{otherwise}\end{cases}\\ &= |\{b \in \{1,\ldots,B\} : i \in \beta_b\}|,\end{aligned}$$

where the first equality follows from algebra and from the fact that $|\mathcal{N}_c| = |\mathcal{N}_{c'}|$, the second equality follows from our construction of $\sigma \in \Sigma$ on line (15), and the third equality follows from the supposition that the equality $|\{b \in \{1,\ldots,B\} : \mathcal{N}_c^k \in \beta_b\}| = |\{b \in \{1,\ldots,B\} : \mathcal{N}_{c'}^k \in \beta_b\}|$ holds for all $k \in \{1, \ldots, |\mathcal{N}_c|\}$. Moreover, we observe for each $b \in \{1, \ldots, B\}$ and each contest $c'' \in \mathcal{C}$ that

$$\begin{aligned}
&|\{\sigma(i) \in \beta_b : i \in \mathcal{N}_{c''}\}| \\
&= \begin{cases} \left|\left\{\sigma\left(\mathcal{N}_{c'}^k\right) : k \in \{1, \ldots, |\mathcal{N}_c|\} \text{ and } \mathcal{N}_{c'}^k \in \beta_b\right\}\right|, & \text{if } c'' = c', \\ \left|\left\{\sigma\left(\mathcal{N}_{c}^k\right) : k \in \{1, \ldots, |\mathcal{N}_c|\} \text{ and } \mathcal{N}_{c}^k \in \beta_b\right\}\right|, & \text{if } c'' = c, \\ |\{\sigma(i) \in \beta_b : i \in \mathcal{N}_{c''}\}|, & \text{otherwise} \end{cases} \\
&= \begin{cases} \left|\left\{\mathcal{N}_{c}^k : k \in \{1, \ldots, |\mathcal{N}_c|\} \text{ and } \mathcal{N}_{c'}^k \in \beta_b\right\}\right|, & \text{if } c'' = c', \\ \left|\left\{\mathcal{N}_{c'}^k : k \in \{1, \ldots, |\mathcal{N}_c|\} \text{ and } \mathcal{N}_{c}^k \in \beta_b\right\}\right|, & \text{if } c'' = c, \\ |\{i \in \beta_b : i \in \mathcal{N}_{c''}\}|, & \text{otherwise} \end{cases} \\
&\le \begin{cases} v_c, & \text{if } c'' = c', \\ v_{c'}, & \text{if } c'' = c, \\ v_{c''}, & \text{otherwise} \end{cases} \\
&= v_{c''},
\end{aligned}$$

where the first equality follows from algebra and from the fact that $|\mathcal{N}_c| = |\mathcal{N}_{c'}|$, the second equality follows from our construction of $\sigma \in \Sigma$ on line (15), the inequality follows from the fact that $\beta_1, \ldots, \beta_B \in \mathscr{B}$, and the third equality follows from the fact that $v_c = v_{c'}$. Combining the above analysis with Theorem 1 from §3.4, we conclude that the equality $T^\sigma(\beta_1, \ldots, \beta_B) = T^*(\beta_1, \ldots, \beta_B)$ is satisfied.

Because we have shown that there exists a non-identity bijection $\sigma \in \Sigma$ that satisfies the equality $T^\sigma(\beta_1, \ldots, \beta_B) = T^*(\beta_1, \ldots, \beta_B)$, we have obtained a contradiction with the fact that $(B, \beta_1, \ldots, \beta_B)$ is a feasible solution of the optimization problem (RO-Σ). Our proof of Lemma 3 is thus complete. □

Proof of Proposition 2. Consider any optimal solution of the optimization problem (RO-Σ), and let that optimal solution be denoted by $(B, \beta_1, \ldots, \beta_B)$. We henceforth assume without loss of generality that for all contests $c < c'$ that satisfy $c \equiv c'$, there exists $k \in \{1, \ldots, |\mathcal{N}_c|\}$ that satisfies

$$\begin{aligned}
\left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_c^{|\mathcal{N}_c|} \in \beta_b\right\}\right| &= \left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_{c'}^{|\mathcal{N}_c|} \in \beta_b\right\}\right| \\
&\vdots \\
\left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_c^{k+1} \in \beta_b\right\}\right| &= \left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_{c'}^{k+1} \in \beta_b\right\}\right| \\
\left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_c^{k} \in \beta_b\right\}\right| &\le \left|\left\{b \in \{1, \ldots, B\} : \mathcal{N}_{c'}^{k} \in \beta_b\right\}\right|.
\end{aligned}$$

This assumption is without loss of generality because the indices of contests that are equivalent can always be permuted to ensure that the vectors $(|\{b \in \{1, \ldots, B\} : \mathcal{N}_c^1 \in \beta_b\}|, \ldots, |\{b \in \{1, \ldots, B\} : \mathcal{N}_c^{|\mathcal{N}_c|} \in \beta_b\}|)$ are lexicographically ordered. Combining that assumption with Lemma 3, our proof of Proposition 2 is complete. □

F.5 Proofs from §5.4

Proof of Theorem 2. Let $\sigma \in \Sigma$ denote a feasible solution of the optimization problem (CUT). For each $k \in \{1, \ldots, K^\sigma\}$, let $\sigma_k : \mathcal{N} \to \mathcal{N}$ be defined for each $c \in \mathcal{C}$ and $i \in \mathcal{N}_c$ by

$$\sigma_k(i) \triangleq \begin{cases} \sigma(i), & \text{if } c \in \mathscr{K}_k^\sigma, \\ i, & \text{if } c \notin \mathscr{K}_k^\sigma. \end{cases}$$

We begin by showing that each of the functions $\sigma_k : \mathcal{N} \to \mathcal{N}$ is a non-identity bijection. Indeed, we observe for each contest $c \notin \mathscr{K}_k^\sigma$ and each candidate $i \in \mathcal{N}_c$ that the equality $\sigma_k^{-1}(i) = i$ holds. Moreover, for each contest $c \in \mathscr{K}_k^\sigma$ and each candidate $i \in \mathcal{N}_c$, it follows from our construction of the undirected graph $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$, from the definition of a connected component, and from the inclusion $\sigma \in \Sigma$ that there exists a contest $c' \in \mathscr{K}_k^\sigma$ and a candidate $i' \in \mathcal{N}_{c'}$ that satisfies $i' \neq i$ and $\sigma_k^{-1}(i) = i'$. Therefore, we have shown for all candidates $i \in \mathcal{N}$ that there exists $i' \in \mathcal{N}$ that satisfies the equality $\sigma_k^{-1}(i') = i$, which concludes our proof that σ_k is a bijection. Moreover, since $\mathscr{K}_k^\sigma$ is nonempty, we have argued that there must exist candidates $i' \neq i$ that satisfy $\sigma_k^{-1}(i) = i'$. Therefore, we conclude that σ_k is a non-identity bijection.

We next show that each of the functions $\sigma_k : \mathcal{N} \to \mathcal{N}$ is a feasible solution of the optimization problem (CUT). Indeed, we have already shown that σ_k is a non-identity bijection, which implies that $\sigma_k \in \Sigma$. Moreover, for each contest $c \in \mathcal{C}$ and each candidate $i \in \mathcal{N}_c$, we observe that

$$\begin{aligned}
&T_i^{\sigma_k}(\beta_1, \ldots, \beta_B) \\
&= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma_k(i) \in \beta_b \text{ and } |\{\sigma_k(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\} \\
&= \begin{cases} \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\}, & \text{if } c \in \mathscr{K}_k^\sigma, \\ \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\{j \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\}, & \text{if } c \notin \mathscr{K}_k^\sigma \end{cases} \\
&= \begin{cases} T_i^\sigma(\beta_1, \ldots, \beta_B), & \text{if } c \in \mathscr{K}_k^\sigma, \\ T_i^*(\beta_1, \ldots, \beta_B), & \text{if } c \notin \mathscr{K}_k^\sigma \end{cases} \\
&= T_i^*(\beta_1, \ldots, \beta_B).
\end{aligned}$$

Indeed, the first equality is the definition of $T_i^{\sigma_k}(\beta_1, \ldots, \beta_B)$. The second equality follows from the definition of σ_k. The third equality follows from the definitions of $T_i^\sigma(\beta_1, \ldots, \beta_B)$ and $T_i^*(\beta_1, \ldots, \beta_B)$. The fourth equality follows from the fact that $\sigma \in \Sigma$ is a feasible solution of the optimization problem (CUT), which implies that the equality $T_i^\sigma(\beta_1, \ldots, \beta_B) = T_i^*(\beta_1, \ldots, \beta_B)$ holds for all candidates $i \in \mathcal{N}$. Our proof that σ_k is a feasible solution of the optimization problem (CUT) is thus complete.

As our final step, we show that line (6) holds. Indeed, we observe for each $k \in \{1, \ldots, K^\sigma\}$, $B \in \mathbb{N}$, $(\beta_1, \ldots, \beta_B) \in \mathscr{B}^B$, $c \in \mathcal{C}$, and $i \in \mathcal{N}_c$ that

$$\begin{aligned}
&T_i^{\sigma_k}(\beta_1, \ldots, \beta_B) \\
&= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma_k(i) \in \beta_b \text{ and } |\{\sigma_k(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\} \\
&= \begin{cases} \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\}, & \text{if } c \in \mathscr{K}_k^\sigma, \\ \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\{j \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\}, & \text{if } c \notin \mathscr{K}_k^\sigma \end{cases} \\
&= \begin{cases} T_i^\sigma(\beta_1, \ldots, \beta_B), & \text{if } c \in \mathscr{K}_k^\sigma, \\ T_i^*(\beta_1, \ldots, \beta_B), & \text{if } c \notin \mathscr{K}_k^\sigma, \end{cases}
\end{aligned} \tag{16}$$

where the first equality is the definition of $T_i^{\sigma_k}(\beta_1, \ldots, \beta_B)$, the second equality follows from the definition of σ_k, and the third equality follows from the definitions of $T_i^\sigma(\beta_1, \ldots, \beta_B)$ and $T_i^*(\beta_1, \ldots, \beta_B)$.

Therefore, we observe that

$$\begin{aligned}
&\bigcup_{k=1}^{K^\sigma} \mathscr{F}\left(\widehat{\Sigma} \cup \{\sigma_k\}\right) \\
&= \bigcup_{k=1}^{K^\sigma} \left(\mathscr{F}\left(\widehat{\Sigma}\right) \cap \bigcup_{B \in \mathbb{N}} \left\{ (\beta_1,\ldots,\beta_B) \in \mathscr{B}^B : T^{\sigma_k}(\beta_1,\ldots,\beta_B) \neq T^*(\beta_1,\ldots,\beta_B) \right\} \right) \\
&= \mathscr{F}\left(\widehat{\Sigma}\right) \cap \bigcup_{B \in \mathbb{N}} \bigcup_{k=1}^{K^\sigma} \left\{ (\beta_1,\ldots,\beta_B) \in \mathscr{B}^B : T^{\sigma_k}(\beta_1,\ldots,\beta_B) \neq T^*(\beta_1,\ldots,\beta_B) \right\} \\
&= \mathscr{F}\left(\widehat{\Sigma}\right) \cap \bigcup_{B \in \mathbb{N}} \bigcup_{k=1}^{K^\sigma} \bigcup_{c \in \mathcal{C}} \bigcup_{i \in \mathcal{N}_c} \left\{ (\beta_1,\ldots,\beta_B) \in \mathscr{B}^B : T_i^{\sigma_k}(\beta_1,\ldots,\beta_B) \neq T_i^*(\beta_1,\ldots,\beta_B) \right\} \\
&= \mathscr{F}\left(\widehat{\Sigma}\right) \cap \bigcup_{B \in \mathbb{N}} \bigcup_{k=1}^{K^\sigma} \bigcup_{c \in \mathscr{K}_k^\sigma} \bigcup_{i \in \mathcal{N}_c} \left\{ (\beta_1,\ldots,\beta_B) \in \mathscr{B}^B : T_i^{\sigma}(\beta_1,\ldots,\beta_B) \neq T_i^*(\beta_1,\ldots,\beta_B) \right\} \\
&= \mathscr{F}\left(\widehat{\Sigma}\right) \cap \bigcup_{B \in \mathbb{N}} \bigcup_{c \in \mathscr{V}^\sigma} \bigcup_{i \in \mathcal{N}_c} \left\{ (\beta_1,\ldots,\beta_B) \in \mathscr{B}^B : T_i^{\sigma}(\beta_1,\ldots,\beta_B) \neq T_i^*(\beta_1,\ldots,\beta_B) \right\} \\
&= \mathscr{F}\left(\widehat{\Sigma}\right) \cap \bigcup_{B \in \mathbb{N}} \bigcup_{c \in \mathcal{C}} \bigcup_{i \in \mathcal{N}_c} \left\{ (\beta_1,\ldots,\beta_B) \in \mathscr{B}^B : T_i^{\sigma}(\beta_1,\ldots,\beta_B) \neq T_i^*(\beta_1,\ldots,\beta_B) \right\} \\
&= \mathscr{F}\left(\widehat{\Sigma}\right) \cap \bigcup_{B \in \mathbb{N}} \left\{ (\beta_1,\ldots,\beta_B) \in \mathscr{B}^B : T^{\sigma}(\beta_1,\ldots,\beta_B) \neq T^*(\beta_1,\ldots,\beta_B) \right\} \\
&= \mathscr{F}\left(\widehat{\Sigma} \cup \{\sigma\}\right).
\end{aligned}$$

Indeed, the first equality follows from the definition of the optimization problem (RO-$\widehat{\Sigma}$). The second and third equalities follow from algebra. The fourth equality follows from line (16). The fifth equality follows from the fact that $\mathscr{K}_1^\sigma, \ldots, \mathscr{K}_{K^\sigma}^\sigma$ are the connected components of the undirected graph $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$, which implies that $\mathscr{K}_1^\sigma \cup \cdots \cup \mathscr{K}_{K^\sigma}^\sigma = \mathscr{V}^\sigma$. The sixth equality follows from the definition of $\mathscr{V}^\sigma$, which implies that the equality $\sigma(i) = i$ is satisfied for all candidates $i \in \cup_{c \in \mathcal{C} \setminus \mathscr{V}^\sigma} \mathcal{N}_c$. The seventh and eighth equalities follow from algebra. Our proof of Theorem 2 is thus complete. □

Proof of Theorem 3. Consider any optimal solution $x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$ of the mixed-integer linear optimization problem (7). Let $\sigma : \mathcal{N} \to \mathcal{N}$ be the function that satisfies the equality $\sigma(i) = j$ if and only if $x_{i,j} = 1$ for all $i, j \in \mathcal{N}$. In this case, it follows from the discussion in §4.2.2 that σ is a non-identity bijection that is a feasible solution for the optimization problem (CUT).

Suppose for the sake of developing a contradiction that the number of connected components of the undirected graph $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$ satisfies $K^\sigma \geq 2$. For each $k \in \{1, \ldots, K^\sigma\}$, let $\sigma_k : \mathcal{N} \to \mathcal{N}$ be defined for each $c \in \mathcal{C}$ and $i \in \mathcal{N}_c$ by

$$\sigma_k(i) \triangleq \begin{cases} \sigma(i), & \text{if } c \in \mathscr{K}_k^\sigma, \\ i, & \text{if } c \notin \mathscr{K}_k^\sigma, \end{cases}$$

where $\mathscr{K}_1^\sigma, \ldots, \mathscr{K}_{K^\sigma}^\sigma \subseteq \mathscr{V}^\sigma$ denote the connected components of the undirected graph $\mathscr{G}^\sigma \equiv (\mathscr{V}^\sigma, \mathscr{E}^\sigma)$ In this case, it follows from Theorem 2 that $\sigma_1, \ldots, \sigma_{K^\sigma}$ are feasible solutions of the optimization problem (CUT). Because $\sigma_1 \in \Sigma$ is a feasible solution of the optimization problem (CUT), we observe that a feasible solution for the mixed-integer linear optimization problem (7) is given by $\bar{x} \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$, which is defined by $\bar{x}_{i,j} \triangleq \mathbb{I}\{\sigma_1(i) = j\}$ for all $i, j \in \mathcal{N}$. We observe that

$$\begin{aligned}
\sum_{i,j \in \mathcal{N}: i \neq j} \bar{x}_{i,j} &= |\{i \in \mathcal{N} : \sigma_1(i) \neq i\}| \\
&= \sum_{c \in \mathscr{K}_1^\sigma} |\{i \in \mathcal{N}_c : \sigma(i) \neq i\}| \\
&< \sum_{c \in \mathcal{C}} |\{i \in \mathcal{N}_c : \sigma(i) \neq i\}| \\
&= \sum_{i,j \in \mathcal{N}: i \neq j} x_{i,j}.
\end{aligned}$$

Indeed, the first equality follows from our construction of $\bar{\sigma}$. The second equality follows from the definition of σ_1. The strict inequality follows from the fact that $K^\sigma \geq 2$, which implies that there exists a candidate $i \notin \cup_{c \in \mathscr{K}_1^\sigma} \mathcal{N}_c$ that satisfies $\sigma(i) \neq i$. The third equality follows from the definition of σ.

In conclusion, we have shown that there exists a feasible solution $\bar{x} \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$ for the mixed-integer linear optimization problem (7) with an objective value that is strictly better than the objective value associated with $x \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$. We thus have a contradiction with the supposition that x is an optimal solution of the mixed-integer linear optimization problem (7), which concludes our proof of Theorem 3. □

F.6 Proofs from §5.5

Proof of Proposition 3. Consider any original ballot style $(\mathcal{N}, \mathcal{C}, \{\mathcal{N}_c\}_{c \in \mathcal{C}}, \{v_c\}_{c \in \mathcal{C}})$, and let the ballot style in which all of the noncompetitive contests from the original ballot style are combined into a single contest be denoted by $(\mathcal{N}, \widetilde{\mathcal{C}}, \{\widetilde{\mathcal{N}}_c\}_{c \in \widetilde{\mathcal{C}}}, \{\widetilde{v}_c\}_{c \in \widetilde{\mathcal{C}}})$. Moreover, consider any $B \in \mathbb{N}$, $\beta_1, \ldots, \beta_B \subseteq \mathcal{N}$, $\sigma \in \Sigma \cup \{*\}$, and $i \in \mathcal{N}$. Finally, let $c \in \mathcal{C}$ denote the contest from the original ballot style that satisfies $i \in \mathcal{N}_c$, and let $\widetilde{c} \in \widetilde{\mathcal{C}}$ denote the contest from the new ballot style that satisfies $i \in \widetilde{\mathcal{N}}_{\widetilde{c}}$. We observe that

$$\begin{aligned}
&\widetilde{T}_i^\sigma(\beta_1, \ldots, \beta_B) \\
&= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } \left|\left\{\sigma(j) \in \beta_b : j \in \widetilde{\mathcal{N}}_{\widetilde{c}}\right\}\right| \leq \widetilde{v}_{\widetilde{c}}\right\} \\
&= \begin{cases} \sum_{b=1}^{B} \mathbb{I}\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\}, & \text{if } \widetilde{c} \neq 0, \\ \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } \left|\left\{\sigma(j) \in \beta_b : j \in \widetilde{\mathcal{N}}_0\right\}\right| \leq \widetilde{v}_0\right\}, & \text{if } \widetilde{c} = 0 \end{cases} \\
&= \begin{cases} \sum_{b=1}^{B} \mathbb{I}\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\}, & \text{if } \widetilde{c} \neq 0, \\ \sum_{b=1}^{B} \mathbb{I}\{\sigma(i) \in \beta_b\}, & \text{if } \widetilde{c} = 0 \end{cases} \\
&= \begin{cases} \sum_{b=1}^{B} \mathbb{I}\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\}, & \text{if } \widetilde{c} \neq 0, \\ \sum_{b=1}^{B} \mathbb{I}\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\}, & \text{if } \widetilde{c} = 0 \end{cases} \\
&= T_i^\sigma(\beta_1, \ldots, \beta_B).
\end{aligned}$$

The first equality is the definition of $\widetilde{T}_i^\sigma(\cdot)$. The second equality follows from the fact that if $\widetilde{c} \neq 0$, then it follows from the construction of the new ballot style that $c = \widetilde{c}$, $\widetilde{\mathcal{N}}_{\widetilde{c}} = \mathcal{N}_c$, and $\widetilde{v}_{\widetilde{c}} = v_c$.

The third equality follows from the fact that $\widetilde{v}_0 = |\widetilde{\mathcal{N}}_0|$. The fourth equality follows from the facts that $i \in \mathcal{N}_c$ and $v_c = |\mathcal{N}_c|$. The fifth equality follows from the definition of $T_i^\sigma(\cdot)$. Our proof of Proposition 3 is thus complete. □

F.7 Proofs from Appendix A

Proof of Proposition 4. Consider a test deck defined by the following equalities:

$$\begin{aligned}
\beta_1 &= \{1\},\\
\beta_2, \beta_3 &= \{2\},\\
\beta_4, \beta_5, \beta_6 &= \{3\},\\
&\vdots\\
\beta_{\frac{N(N-1)}{2}+1}, \ldots, \beta_{\frac{N(N+1)}{2}} &= \{N\}.
\end{aligned}$$

We observe that the above test deck consists of $B = N(N+1)/2$ filled-out ballots. Moreover, it follows from the fact that $v_c \geq 1$ for all contests $c \in \mathcal{C}$ that the above filled-out ballots satisfy $\beta_1, \ldots, \beta_B \in \mathscr{B}$.

It remains for us to show that the test deck defined above satisfies the constraints of the optimization problem (RO-Σ). Indeed, we observe that the test deck satisfies the equality $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = i$ for each candidate $i \in \mathcal{N} \equiv \{1, \ldots, N\}$. Furthermore, we recall for each non-identity bijection $\sigma \in \Sigma$ that there must exist a candidate $i \in \mathcal{N}$ that satisfies $\sigma(i) \neq i$. Therefore, we conclude for each non-identity bijection $\sigma \in \Sigma$ that there exists a candidate $i \in \mathcal{N}$ that satisfies $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \neq |\{b \in \{1, \ldots, B\} : \sigma(i) \in \beta_b\}|$, which together with Corollary 1 implies that the test deck satisfies the constraints of the optimization problem (RO-Σ). Our proof of Proposition 4 is thus complete. □

Proof of Proposition 5. We begin by showing that the optimal objective value of the optimization problem (11) is greater than or equal to the optimal objective value of the mixed-integer linear optimization problem (12). Indeed, let $(B, \beta_1, \ldots, \beta_B)$ denote an optimal solution for the optimization problem (11). This optimal solution assigns each candidate in $\mathcal{N}$ some distinct number of votes. We assume without loss of generality that the solution assigns each candidate some distinct number of votes between 1 and $|\mathcal{N}|$; if this property does not hold for a given solution, one can simply omit votes for the candidates receiving more than $|\mathcal{N}|$ votes to achieve this property without requiring any additional ballots.

From this optimal test deck, we construct a binary vector $\gamma \in \{0,1\}^{\mathcal{C} \times \mathcal{N}}$ that is defined for each $c \in \mathcal{C}$ and $g \in \mathcal{N}$ as

$$\gamma_{c,g} \triangleq \mathbb{I}\left\{\text{there exists } i \in \mathcal{N}_c \text{ such that } |\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = g\right\}.$$

In the following bullet points, we show that the integer $B \in \mathbb{N}$ and the binary vector $\gamma \in \{0,1\}^{\mathcal{C} \times \mathcal{N}}$ satisfy each of the constraints of the mixed-integer linear optimization problem (12):

- We first show that B, γ satisfies constraint (12b). Indeed, we observe for each contest $c \in \mathcal{C}$

that

$$\begin{aligned}\sum_{g \in \mathcal{N}} \gamma_{c,g} &= \sum_{g \in \mathcal{N}} \mathbb{I}\left\{\text{there exists } i \in \mathcal{N}_c \text{ such that } \left|\{b \in \{1,\ldots,B\} : i \in \beta_b\}\right| = g\right\} \\ &= \sum_{i \in \mathcal{N}_c} \sum_{g \in \mathcal{N}} \mathbb{I}\left\{\left|\{b \in \{1,\ldots,B\} : i \in \beta_b\}\right| = g\right\} \\ &= \sum_{i \in \mathcal{N}_c} 1 \\ &= |\mathcal{N}_c|.\end{aligned}$$

The first equality follows from the definition of $\gamma_{c,g}$. The second equality follows from the fact that $\beta_1,\ldots,\beta_B$ satisfies constraint (11b). The third equality follows from the fact that $\beta_1,\ldots,\beta_B$ satisfies constraint (11c) and from our earlier assumption that the inclusion $|\{b \in \{1,\ldots,B\} : i \in \beta_b\}| \in \mathcal{N}$ holds for all candidates $i \in \mathcal{N}$. The fourth equality follows from algebra.

- We next show that B, γ satisfies constraint (12c). Indeed, we observe for each $g \in \mathcal{N}$ that

$$\begin{aligned}\sum_{c \in \mathcal{C}} \gamma_{c,g} &= \sum_{c \in \mathcal{C}} \mathbb{I}\left\{\text{there exists } i \in \mathcal{N}_c \text{ such that } \left|\{b \in \{1,\ldots,B\} : i \in \beta_b\}\right| = g\right\} \\ &= \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_c} \mathbb{I}\left\{\left|\{b \in \{1,\ldots,B\} : i \in \beta_b\}\right| = g\right\} \\ &= \sum_{i \in \mathcal{N}} \mathbb{I}\left\{\left|\{b \in \{1,\ldots,B\} : i \in \beta_b\}\right| = g\right\} \\ &= 1.\end{aligned}$$

The first equality follows from the definition of $\gamma_{c,g}$. The second equality follows from the fact that $\beta_1,\ldots,\beta_B$ satisfies constraint (11b). The third equality follows from algebra. The fourth equality follows from our earlier assumption that $|\{b \in \{1,\ldots,B\} : i \in \beta_b\}| \in \mathcal{N}$ for all candidates $i \in \mathcal{N}$, which together with the fact that $\beta_1,\ldots,\beta_B$ satisfies constraint (11c) implies that there must exist exactly one candidate $i \in \mathcal{N}$ that satisfies the equality $|\{b \in \{1,\ldots,B\} : i \in \beta_b\}| = g$.

- We next show that B, γ satisfies constraint (12d). Indeed, we observe for each contest $c \in \mathcal{C}$

that

$$\begin{aligned}
&\frac{1}{v_c} \sum_{g \in \mathcal{N}} g \gamma_{c,g} \\
&= \frac{1}{v_c} \sum_{g \in \mathcal{N}} g \mathbb{I} \left\{ \text{there exists } i \in \mathcal{N}_c \text{ such that } |\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = g \right\} \\
&= \frac{1}{v_c} \sum_{i \in \mathcal{N}_c} \sum_{g \in \mathcal{N}} g \mathbb{I} \left\{ |\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = g \right\} \\
&= \frac{1}{v_c} \sum_{i \in \mathcal{N}_c} |\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \\
&= \frac{1}{v_c} \sum_{b=1}^{B} |\mathcal{N}_c \cap \beta_b| \\
&\leq \frac{1}{v_c} \sum_{b=1}^{B} v_c \\
&= B.
\end{aligned}$$

The first equality follows from the definition of γ. The second equality follows from the fact that $\beta_1, \ldots, \beta_B$ satisfies constraint (11b). The third equality follows from the fact that $\beta_1, \ldots, \beta_B$ satisfies constraint (11c) and from our earlier assumption that the inclusion $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \in \mathcal{N}$ holds for all candidates $i \in \mathcal{N}$. The fourth equality follows from algebra. The inequality follows from the fact that $\beta_1, \ldots, \beta_B \in \mathscr{B}$. The fifth equality follows from algebra.

- Finally, we show that B, γ satisfies constraint (12e). Indeed, it follows from the fact that $\beta_1, \ldots, \beta_B$ satisfies constraint (11c) and from our assumption that $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| \in \mathcal{N}$ for all candidates $i \in \mathcal{N}$ that there must exist a candidate $i \in \mathcal{N}$ that satisfies the equality $|\{b \in \{1, \ldots, B\} : i \in \beta_b\}| = N$. Therefore, we conclude that the inequality $B \geq N$ must be satisfied.

In summary, we have shown in the above bullet points that the integer $B \in \mathbb{N}$ and the binary vector $\gamma \in \{0,1\}^{\mathcal{C} \times \mathcal{N}}$ is a feasible but possibly sub-optimal solution for the mixed-integer linear optimization problem (12). Because $(B, \beta_1, \ldots, \beta_B)$ is an optimal solution for the optimization problem (11), our proof that the optimal objective value of the optimization problem (11) is greater than or equal to the optimal objective value of the mixed-integer linear optimization problem (12) is thus complete.

It remains for us to show that the optimal objective value of the mixed-integer linear optimization problem (12) is greater than or equal to the optimal objective value of the optimization problem (11). To show this, let $B \in \mathbb{N}$ and $\gamma \in \{0,1\}^{\mathcal{C} \times \mathcal{N}}$ denote any optimal solution of the mixed-integer linear optimization problem (12). Moreover, let $\pi : \mathcal{N} \to \mathcal{N}$ denote the function that satisfies the equality $\pi(i) = \sum_{g \in \mathcal{N}} g \gamma_{c,g}$ for all contests $c \in \mathcal{C}$ and candidates $i \in \mathcal{N}_c$. It follows from the fact that B, γ is a feasible solution for the mixed-integer linear optimization problem (12) that the function π is a bijection. Given the bijection π, we now construct a test deck $(\beta_1, \ldots, \beta_B)$ using the following procedure:

```
$\beta_1, \dots, \beta_B \leftarrow \emptyset$
for $c \in \mathcal{C}$ do
    $b \leftarrow 1$
    for $i \in \mathcal{N}_c$ do
        for $\ell \in \{1, \dots, \pi(i)\}$ do
            $\beta_b \leftarrow \beta_b \cup \{i\}$
            $b \leftarrow (b \mod B) + 1$
        end for
    end for
end for
```

The procedure begins by initializing B blank ballots. Then, for each contest $c \in \mathcal{C}$, the procedure iterates through the ballots and adds the candidates to the ballots. It follows from the fact that $B \geq N$ and from the fact that $\pi(j) \in \mathcal{N}$ for all candidates $j \in \mathcal{N}$ that each candidate $i \in \mathcal{N}_c$ will be selected by this procedure by $\pi(i)$ different ballots. Moreover, it follows from the fact that $B \geq \frac{1}{v_c} \sum_{i \in \mathcal{N}_c} \pi(i)$ that the procedure will select no more than v_c of the targets from $\mathcal{N}_c$ in any ballot. Therefore, we conclude that the procedure will output a test deck that satisfies $\beta_1, \dots, \beta_B \in \mathscr{B}$ as well as satisfies all of the constraints of the optimization problem (11). Because we have shown that any optimal solution for the mixed-integer linear optimization problem (12) can be transformed into a feasible solution for the optimization problem (11) with the same objective value, we conclude that the optimal objective value of the mixed-integer linear optimization problem (12) must be greater than or equal to the optimal objective value of the optimization problem (11). Our proof of Proposition 5 is thus complete. □

F.8 Proofs from Appendix B

Proof of Proposition 6. Construct a deck of ballots $\beta_1, \dots, \beta_N$ such that for each $i \in \mathcal{N}$, $\beta_i = \{i\}$. It holds that $T^*_i(\beta_1, \dots, \beta_N) = 1$ for each $i \in \mathcal{N}$, since only ballot β_i is interpreted as containing a vote for candidate i. It also holds for any $\sigma \in \Sigma$ and candidate $i \in \mathcal{N}$ that $T^\sigma_i(\beta_1, \dots, \beta_N) = 1$, since only ballot $\beta_{\sigma(i)}$ is interpreted as containing a vote for candidate i. Thus, we conclude that $T^\sigma(\beta_1, \dots, \beta_B) = T^*(\beta_1, \dots, \beta_B)$ and $T^*_i(\beta_1, \dots, \beta_N) = 1 \geq 1$ for all $i \in \mathcal{N}$. □

Proof of Theorem 4. Recall that each $\sigma \in \Sigma$ can be interpreted as a permutation on $\mathcal{N}$, which implies that it can be decomposed into a number of cycles with disjoint sets of elements.[15] Let the set of elements in each of the K cycles be denoted $\mathcal{O}_1, \dots, \mathcal{O}_K$. We now construct a test deck $\beta_1, \dots, \beta_B$ such that $B \triangleq K(K+1)/2$ and

$$\begin{aligned}
\beta_1 &\triangleq \mathcal{O}_1,\\
\beta_2, \beta_3 &\triangleq \mathcal{O}_2\\
\beta_4, \beta_5, \beta_6 &\triangleq \mathcal{O}_3\\
&\vdots\\
\beta_{\frac{K(K-1)}{2}+1}, \dots, \beta_{\frac{K(K+1)}{2}} &\triangleq \mathcal{O}_K.
\end{aligned}$$

Because the bidirectional implication $\sigma^n(i) \in \mathcal{N}_c \iff \sigma^n(i) = i$ holds for all contests $c \in \mathcal{C}$, candidates $i \in \mathcal{N}_c$, and integers $n \in \mathbb{N}$, we know that each set $\mathcal{O}_k$ includes at most one candidate

[15]We say that $i, j \in \mathcal{N}$ are in the same cycle if and only if there exists an integer $k \in \mathbb{N}$ that satisfies $\sigma^k(i) = j$.

from each contest. This means at most one candidate from each contest is marked on each ballot, thereby implying that $\beta_1, \ldots, \beta_B \in \mathscr{B}$. Moreover, for each candidate $i \in \mathcal{O}_k$, we observe that

$$\begin{aligned} T_i^*(\beta_1, \ldots, \beta_B) &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \leq v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b\right\} \\ &= k. \end{aligned}$$

The first equality is the definition of $T_i^*(\cdot)$. The second equality follows because $\beta_1, \ldots, \beta_B \in \mathscr{B}$. The third equality follows from the fact that the test deck has been constructed to contain k ballots that vote for the candidates in $\mathcal{O}_k$.

We conclude by showing that the voting machine with mapping σ gives the correct output for each candidate $i \in \mathcal{N}_c$ in each contest $c \in \mathcal{C}$:

$$\begin{aligned} T_i^\sigma(\beta_1, \ldots, \beta_B) &= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\{j \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \leq v_c\right\} \\ &= T_i^*(\beta_1, \ldots, \beta_B). \end{aligned}$$

The first equality is the definition of $T^\sigma(\cdot)$. The second equality holds because each ballot β_b marks every candidate that falls in the same cycle under σ, which implies that $\sigma(j) \in \beta_b \iff j \in \beta_b$ for all $j \in \mathcal{N}_c$. The third equality follows from algebra. The fourth equality is the definition of $T^*(\cdot)$. Our proof of Theorem 4 is thus complete. □

F.9 Proofs from Appendix D

Proof of Proposition 7. Let $\bar{\beta} \triangleq \mathcal{N}$ be the ballot that votes for every target, and let $(B, \beta_1, \ldots, \beta_B)$ denote a feasible solution for the optimization problem (RO-Σ). For each candidate $i \in \mathcal{N}$, we know that the reported vote total under any incorrect mapping $\sigma \in \Sigma$ is as follows, where $c \in \mathcal{C}$ is the contest containing candidate i:

$$\begin{aligned} T_i^\sigma(\beta_1, \ldots, \beta_B, \bar{\beta}) &= \left(\sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c\right\} \right) \\ &\quad + \mathbb{I}\left\{\sigma(i) \in \bar{\beta} \text{ and } \left|\left\{\sigma(j) \in \bar{\beta} : j \in \mathcal{N}_c\right\}\right| \leq v_c\right\} \\ &= T_i^\sigma(\beta_1, \ldots, \beta_B) + \mathbb{I}\left\{\sigma(i) \in \bar{\beta} \text{ and } \left|\left\{\sigma(j) \in \bar{\beta} : j \in \mathcal{N}_c\right\}\right| \leq v_c\right\} \\ &= T_i^\sigma(\beta_1, \ldots, \beta_B) + \mathbb{I}\left\{|\mathcal{N}_c| \leq v_c\right\}. \end{aligned}$$

The first two equalities follow from the definition of $T_i^\sigma(\cdot)$, and the third equality follows from the fact that $\bar{\beta} = \mathcal{N}$. The reported vote total on a properly functioning voting machine, meanwhile, is given by the following:

$$\begin{aligned} T_i^*(\beta_1,\dots,\beta_B,\bar\beta) &= \left(\sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \le v_c\right\}\right) \\ &\quad + \mathbb{I}\left\{i \in \bar\beta \text{ and } \left|\bar\beta \cap \mathcal{N}_c\right| \le v_c\right\} \\ &= T_i^*(\beta_1,\dots,\beta_B) + \mathbb{I}\left\{i \in \bar\beta \text{ and } \left|\bar\beta \cap \mathcal{N}_c\right| \le v_c\right\} \\ &= T_i^*(\beta_1,\dots,\beta_B) + \mathbb{I}\left\{|\mathcal{N}_c| \le v_c\right\}. \end{aligned}$$

The first two equalities follow from the definition of $T_i^*(\cdot)$, and the third equality again follows from the fact that $\bar\beta = \mathcal{N}$.

We observe that because $(B, \beta_1, \dots, \beta_B)$ is a feasible solution for the optimization problem (RO-Σ), it must be the case that $T^\sigma(\beta_1,\dots,\beta_B) \neq T^*(\beta_1,\dots,\beta_B)$ for all $\sigma \in \Sigma$. This means that for each such σ the resulting vectors must differ in at least one position; that is, there must exist some $i \in \mathcal{N}$ such that $T_i^\sigma(\beta_1,\dots,\beta_B) \neq T_i^*(\beta_1,\dots,\beta_B)$. For this i, we can conclude the following:

$$\begin{aligned} T_i^\sigma(\beta_1,\dots,\beta_B,\bar\beta) &= T_i^\sigma(\beta_1,\dots,\beta_B) + \mathbb{I}\left\{|\mathcal{N}_c| \le v_c\right\} \\ &\neq T_i^*(\beta_1,\dots,\beta_B) + \mathbb{I}\left\{|\mathcal{N}_c| \le v_c\right\} \\ &= T_i^*(\beta_1,\dots,\beta_B,\bar\beta). \end{aligned}$$

The two equalities follow from the chain of equalities derived above, and the non-equality follows from the fact that $T_i^\sigma(\beta_1,\dots,\beta_B) \neq T_i^*(\beta_1,\dots,\beta_B)$ for the given $i \in \mathcal{N}$. We have therefore shown that there exists some $i \in \mathcal{N}$ for each $\sigma \in \Sigma$ such that $T_i^\sigma(\beta_1,\dots,\beta_B,\bar\beta) \neq T_i^*(\beta_1,\dots,\beta_B,\bar\beta)$. This means the vectors $T^\sigma(\beta_1,\dots,\beta_B,\bar\beta)$ and $T^*(\beta_1,\dots,\beta_B,\bar\beta)$ differ in at least one position, so Proposition 7 is proven. □

Proof of Proposition 8. Let $(B, \beta_1, \dots, \beta_B)$ denote a feasible solution to the optimization problem (RO-Σ), let $\sigma \in \Sigma$, and let $\tilde\beta \subseteq \mathcal{N}$ denote any filled-out ballot that satisfies the following equality for each contest $c \in \mathcal{C}$:

$$|\tilde\beta \cap \mathcal{N}_c| = \begin{cases} v_c + 1, & \text{if } |\mathcal{N}_c| > v_c, \\ 0, & \text{otherwise.} \end{cases}$$

If there exists $b \in \mathcal{B}$ and $c \in \mathcal{C}$ such that $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| > v_c$, then our proof is complete. Therefore, we assume for the rest of the proof of Proposition 8 that the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c$ holds for all $b \in \mathcal{B}$ and $c \in \mathcal{C}$. This allows us to determine that the following holds:

$$\begin{aligned} \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_c} T_i^*(\beta_1,\dots,\beta_B) &= \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_c} \sum_{b=1}^{B} \mathbb{I}\{i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \le v_c\} \\ &= \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_c} \sum_{b=1}^{B} \mathbb{I}\{i \in \beta_b\} \\ &= \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_c} \sum_{b=1}^{B} \mathbb{I}\{\sigma(i) \in \beta_b\} \\ &= \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_c} \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\} \\ &= \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_c} T_i^\sigma(\beta_1,\dots,\beta_B). \end{aligned} \tag{17}$$

The first equality is the definition of $T_i^*(\cdot)$. The second equality holds because $\beta_1, \ldots, \beta_B \in \mathscr{B}$. The third equality holds because σ is a bijection over $\mathcal{N}$, so the transformation only permutes the order in which terms are added to the sum. The fourth equality holds due to our assumption that $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c$. The fifth equality is the definition of $T_i^\sigma(\cdot)$.

It follows from the fact that $(B, \beta_1, \ldots, \beta_B)$ is a feasible solution to the optimization problem (RO-Σ) and from the fact that $\sigma \in \Sigma$ that $T^*(\beta_1, \ldots, \beta_B) \neq T^\sigma(\beta_1, \ldots, \beta_B)$. It follows from this fact and from the equality derived on line (17) that there must exist a candidate $i \in \mathcal{N}_c$ in some contest $c \in \mathcal{C}$ that satisfies the strict inequality $T_i^*(\beta_1, \ldots, \beta_B) < T_i^\sigma(\beta_1, \ldots, \beta_B)$. For this candidate i, the following must hold:

$$\begin{aligned}
T_i^*(\beta_1, \ldots, \beta_B, \tilde{\beta}) &= \left(\sum_{b=1}^{B} \mathbb{I}\left\{ i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \leq v_c \right\} \right) \\
&\quad + \mathbb{I}\left\{ i \in \tilde{\beta} \text{ and } \left|\tilde{\beta} \cap \mathcal{N}_c\right| \leq v_c \right\} \\
&= \left(\sum_{b=1}^{B} \mathbb{I}\left\{ i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \leq v_c \right\} \right) \\
&= T_i^*(\beta_1, \ldots, \beta_B) \\
&< T_i^\sigma(\beta_1, \ldots, \beta_B) \\
&= \left(\sum_{b=1}^{B} \mathbb{I}\left\{ \begin{array}{r} \sigma(i) \in \beta_b \text{ and} \\ |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c \end{array} \right\} \right) \\
&\leq \left(\sum_{b=1}^{B} \mathbb{I}\left\{ \begin{array}{r} \sigma(i) \in \beta_b \text{ and} \\ |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c \end{array} \right\} \right) \\
&\quad + \mathbb{I}\left\{ \begin{array}{r} \sigma(i) \in \beta_b \text{ and} \\ |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c \end{array} \right\} \\
&= T_i^\sigma(\beta_1, \ldots, \beta_B, \tilde{\beta}).
\end{aligned}$$

The first equality is the definition of $T_i^*(\cdot)$. The second equality holds because the construction of the filled-out ballot $\tilde{\beta}$ implies that either $|\tilde{\beta} \cap \mathcal{N}_c| = v_c + 1$ or $i \notin \tilde{\beta}$. The third equality is the definition of $T_i^*(\cdot)$. The first inequality follows for candidate i by earlier reasoning. The fourth equality is the definition of $T_i^\sigma(\cdot)$. The second inequality holds because $\mathbb{I}\{\cdot\}$ is non-negative. The fifth equality is the definition of $T_i^\sigma(\cdot)$.

In summary, we have shown that if the inequality $|\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \leq v_c$ holds for all $b \in \mathcal{B}$ and $c \in \mathcal{C}$, then there must exist a candidate $i \in \mathcal{N}$ that satisfies $T_i^*(\beta_1, \ldots, \beta_B, \tilde{\beta}) \neq T_i^\sigma(\beta_1, \ldots, \beta_B, \tilde{\beta})$. Our proof of Proposition 8 is thus complete. □

F.10 Proofs from Appendix E

Proof of Lemma 4. Let $(B, \beta_1, \ldots, \beta_B)$ be an optimal solution for the optimization problem (RO-Σ) for a ballot style parameterized by the tuple $(\mathcal{N}, \mathcal{C}, \{\mathcal{N}_c\}_{c \in \mathcal{C}}, \{v_c\}_{c \in \mathcal{C}})$. Let $\bar{\mathcal{C}} \subset \mathcal{C}$ be a subset of that ballot style's contests which we are removing from the ballot style, and let $\bar{\mathcal{N}} \triangleq \bigcup_{c \in \bar{\mathcal{C}}} \mathcal{N}_c$ be the candidates in those contests. Define $\mathcal{C}' \triangleq \mathcal{C} \setminus \bar{\mathcal{C}}$ and $\mathcal{N}' \triangleq \mathcal{N} \setminus \bar{\mathcal{N}}$ as the contests and candidates left over when the subsets $\bar{\mathcal{C}}$ and $\bar{\mathcal{N}}$ are removed. Consider the ballot style created when the candidates $\bar{\mathcal{N}}$ and contests $\bar{\mathcal{C}}$ are removed, which is parameterized by the tuple $(\mathcal{N}', \mathcal{C}', \{\mathcal{N}_c\}_{c \in \mathcal{C}'}, \{v_c\}_{c \in \mathcal{C}'})$. Define Σ' as the set of non-identity bijections over $\mathcal{N}'$; that is, allow it to be the set of possible mappings for this new ballot style.

Consider some particular $\sigma' \in \Sigma'$, and let the extension of this mapping to the domain $\mathcal{N}$ be defined for each candidate $i \in \mathcal{N}$ as

$$\sigma(i) \triangleq \begin{cases} \sigma'(i), & \text{if } i \in \mathcal{N}', \\ i, & \text{if } i \in \bar{\mathcal{N}}. \end{cases}$$

We first observe for each candidate $i \in \bar{\mathcal{N}}$ that

$$\begin{aligned} T_i^\sigma(\beta_1, \ldots, \beta_B) &= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\{j \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \le v_c\right\} \\ &= T_i^*(\beta_1, \ldots, \beta_B). \end{aligned}$$

The first equality holds by the definition of $T_i^\sigma(\cdot)$. The second equality holds because $\sigma(i) = i$ for all $i \in \bar{\mathcal{N}}$. The third equality follows from algebra. The fourth equality follows from the definition of $T_i^\sigma(\cdot)$.

It follows from the fact that $(B, \beta_1, \ldots, \beta_B)$ is a feasible solution for the optimization problem (RO-Σ) that that $T^\sigma(\beta_1, \ldots, \beta_B) \neq T^*(\beta_1, \ldots, \beta_B)$. With the equality derived above, this means that there must exist a candidate $i \in \mathcal{N}'$ that satisfies $T_i^\sigma(\beta_1, \ldots, \beta_B) \neq T_i^*(\beta_1, \ldots, \beta_B)$. Take that candidate i and let $c \in \mathcal{C}'$ be the contest that satisfies $i \in \mathcal{N}_c$. Then, it holds that

$$\begin{aligned} T_i^*(\beta_1 \setminus \bar{\mathcal{N}}, \ldots, \beta_B \setminus \bar{\mathcal{N}}) &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \setminus \bar{\mathcal{N}} \text{ and } \left|(\beta_b \setminus \bar{\mathcal{N}}) \cap \mathcal{N}_c\right| \le v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } \left|(\beta_b \setminus \bar{\mathcal{N}}) \cap \mathcal{N}_c\right| \le v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{i \in \beta_b \text{ and } |\beta_b \cap \mathcal{N}_c| \le v_c\right\} \\ &= T_i^*(\beta_1, \ldots, \beta_B) \\ &\neq T_i^\sigma(\beta_1, \ldots, \beta_B) \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma(i) \in \beta_b \text{ and } |\{\sigma(j) \in \beta_b : j \in \mathcal{N}_c\}| \le v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma'(i) \in \beta_b \text{ and } \left|\{\sigma'(j) \in \beta_b : j \in \mathcal{N}_c\}\right| \le v_c\right\} \\ &= \sum_{b=1}^{B} \mathbb{I}\left\{\sigma'(i) \in \beta_b \setminus \bar{\mathcal{N}} \text{ and } \left|\{\sigma'(j) \in \beta_b \setminus \bar{\mathcal{N}} : j \in \mathcal{N}_c\}\right| \le v_c\right\} \\ &= T_i^{\sigma'}(\beta_1 \setminus \bar{\mathcal{N}}, \ldots, \beta_B \setminus \bar{\mathcal{N}}). \end{aligned}$$

The first equality holds by the definition of $T_i^*(\cdot)$. The second equality holds because $i \notin \bar{\mathcal{N}}$. The third equality holds because $\bar{\mathcal{N}} \cap \mathcal{N}_c = \emptyset$. The fourth equality holds by the definition of $T_i^*(\cdot)$. The non-equality follows from our choice of i. The fifth equality holds by the definition of $T_i^{\sigma}(\cdot)$. The sixth equality holds because $\sigma(i) = \sigma'(i)$ for all $i \in \mathcal{N}'$. The seventh equality holds because σ' has a range which excludes $\bar{\mathcal{N}}$. The eighth equality is the definition of $T_i^{\sigma'}(\cdot)$.

In summary, we have shown for each $\sigma' \in \Sigma'$ that there exists a candidate $i \in \mathcal{N}'$ that satisfies $T_i^*(\beta_1 \setminus \bar{\mathcal{N}}, \ldots, \beta_B \setminus \bar{\mathcal{N}}) \neq T_i^{\sigma'}(\beta_1 \setminus \bar{\mathcal{N}}, \ldots, \beta_B \setminus \bar{\mathcal{N}})$. This fact, along with the observation that $\beta_1 \setminus \bar{\mathcal{N}}, \ldots, \beta_B \setminus \bar{\mathcal{N}} \in \mathscr{B}$ since $\beta_1, \ldots, \beta_B \in \mathscr{B}$, allows us to conclude that $(B, \beta_1 \setminus \bar{\mathcal{N}}, \ldots, \beta_B \setminus \bar{\mathcal{N}})$ is a feasible solution to the optimization problem (RO-Σ) for the ballot style parameterized by the tuple $(\mathcal{N}', \mathcal{C}', \{\mathcal{N}_c\}_{c \in \mathcal{C}'}, \{v_c\}_{c \in \mathcal{C}'})$. $\square$