5-30-2008

A Comparative Study of the Robustness of Voting Systems Under Various Models of Noise

Derek M. Shockey

A Comparative Study of the Robustness of Voting Systems Under Various Models of Noise

Derek M. Shockey
30 May 2008

Chair: Christopher M. Homan
Reader: Piotr Faliszewski
Observer: Charles Border

Contents

Abstract

While the study of election theory is not a new field in and of itself, recent research has applied various concepts in computer science to the study of social choice theory, which includes election theory. From a security perspective, it is pertinent to investigate how stable election systems are in the face of noise, disruption, and manipulation. Recently, work related to computational election systems has also been of interest to artificial intelligence researchers, where it is incorporated into the decision-making processes of distributed systems. The quantitative analysis of a voting rule's resistance to noise is the *robustness*, the probability of how likely the outcome of the election is to change given a certain amount of noise. Prior research has studied the robustness of voting rules under very small amounts of noise, e.g. swapping the ranking of two adjacent candidates in one vote. Our research expands upon this previous work by considering a more disruptive form of noise: an arbitrary reordering of an entire vote. Given k noise disruptions, we determine how likely the election is to remain unchanged (the k-robustness) by relating the k-robustness to the 1-robustness. We can thereby provide upper and/or lower bounds on the robustness of voting rules; specifically, we examine five well-established rules: scoring rules (a general class of rules, containing Borda, plurality, and veto, among others), Copeland, Maximin (also known as Minimax or Simpson–Kramer), Bucklin, and plurality with runoff.

Acknowledgements

I would like to thank my parents and my sister for their enduring and seemingly boundless support in this and all of my endeavors. I would also like to thank my advisor, Chris Homan, for the many, many hours spent working with me on this paper, from selecting a topic all the way through the very end.

Chapter 1

Introduction

An election is generally defined as a collective decision-making process, conducted by casting individual votes to express a preference. Using a predefined rule, votes are tabulated in some manner, and a winner (or winners) is determined. Mathematically speaking, a voting rule is simply a function that outputs a winner given an input of votes. It is common to think of voting as simply choosing a single candidate, but more generally, a vote is an ordered list of all the candidates, expressing preference by relative rank. The selection of a single candidate is actually a ranking in which that candidate is first, and all of the remaining candidates are in a tie for last place.

Most people are probably familiar with elections in political settings, in which the citizens of a country, state, or other municipality vote to elect governmental leaders and representatives. Perhaps the most common or recognizable form of political election is a single-winner election, in which the scoring rule ultimately outputs exactly one winner. This is most obviously used when there is only one position to fill, such as a head of government. It is also common, however, to elect members to legislative bodies of government via single-winner elections by assigning each seat to a multimember constituency (a specific district or region), as is done in both houses of the United States Congress and the lower houses of Parliament in the United Kingdom and Canada. In these cases, a voter is casting a preference for a specific individual.

Multiple-winner elections are also common for electing officials to multiple-member legislative bodies, usually by proportionally distributing seats based on the number of votes for each party (rather than votes for a specific candidate). This system is used for the lower house of parliament in France (the French National Assembly) and both houses of parliament in Italy. The lower house of the German parliament (the Bundestag) is elected half by a proportional multiple-winner election, and half by single-winner plurality elections. There are also non-proportional multiple-winner voting systems, which usually fill the n positions with the top n plurality winners, called bloc voting or plurality-at-large. Generally, a bloc voting system is used for smaller governing bodies, such as a council or a board, though there are instances of legislative bodies being elected by non-proportional voting systems.

The above examples of political elections are founded on a basic principle of "one person, one vote," giving each citizen an equal say. Outside of the world of politics, this is rarely the case. In corporate elections, for example, each shareholder may vote on issues such as who serves on the board of directors, approval of sales and acquisitions, etc., but the votes of a shareholder are weighted by the number of shares owned. This allows any person or entity owning a majority of shares to retain control of the company, as the majority shareholder cannot be outvoted. Furthermore, corporations can issue different classes of shares with different voting rights. This could allow for a non-majority shareholder to still retain the majority of voting rights, and therefore retain control of the corporation.

Since the 13th century, mathematicians have studied methods for conducting elections, which allow a group of voters to express their opinions on a given set of candidates or issues. It would be reasonable to say that the outcome of an election must best represent the aggregate preferences of the voters, or the election has essentially failed its purpose. The study of elections and voting rules has evolved into the field of preference aggregation, a major component of social choice theory.

The general population, especially in the United States, may only be familiar with single vote election systems that are common in modern democratic governments. In practical terms, however, voters often have opinions about many or all candidates, not just their first choice candidate. While many common voting rules consider only the top-ranked candidate, there exist more complex voting systems that take the rankings of every candidate into

account. Resulting from extensive study of social choice mechanisms, many alternative election systems have been developed based on entirely different and more complex mathematical functions that weight the rankings of all the candidates; we will explore several of these, including Copeland, Bucklin, and Maximin.

Most likely, the complexities of these election systems have prevented their widespread use in political elections, even though they may yield a winner who, in some respects, more accurately represents the preferences of the voters. The concerns of practicality are greatly mitigated in an electronic context, however, and as a result the study of preference aggregation has become increasingly relevant in multiagent scenarios and important to the field of distributed artificial intelligence.

Conducting preference aggregation in an electronic context, especially when networked, obviously presents a variety of security issues, many of which have arisen during the deployment of electronic voting machines for use in political elections. The alteration of the aggregate data, whether intentional and malicious or accidental, clearly has the potential for enormous impact on modern society. While the threat pertaining to decision making in artificial intelligence may not seem as serious as in political elections, in both cases there are opportunities to compromise an entire system that utilizes these decision-making mechanisms.

One good measure to determine if a voting rule could be implemented practically in these contexts is *robustness*. The measure of robustness is the quantitative analysis of the probability that the outcome of an election will not change when the election data is altered by a given amount of noise. The noise could in practice be either intentional manipulation of the data, or unintentional random corruption. The measure of robustness is in some ways applicable to both types, but the resistance of voting protocols to manipulation is an entire topic in and of itself [4, 6, 10, 13, 16, 29] that is far beyond the scope of our research. We will instead focus our efforts on the notion of robustness in the face of random noise.

Recently, Procaccia et al. [27] studied the robustness of an election under noise in the form of a swaps between two adjacent candidates in a single voter's ordinal preference list. They call this form of noise an *elementary transposition*. It is straightforward to see that an elementary transposition represents the least significant possible change to a preference profile. This

type of noise would ostensibly be random and unintentional. Procaccia et al. defines the term 1-robustness as being the probability the the outcome of an election remains unchanged after a single fault (one randomly chosen elementary transposition), and k-robustness as the probability of robustness given multiple (k) faults. Comprising the bulk of the research in Procaccia et al. are the upper and/or lower bounds in terms of k-robustness for five different well-established election rules: scoring rules, Copeland, Maximin, Bucklin, and plurality with runoff. The bounds are all strictly worst case, although we discovered in our own work that it is very difficult to prove what specific distributions are actually the worst for a given election rule.

Our intention is to expand upon this research using more severe types of noise and to study their effect on the outcomes of the same five election rules. Procaccia et al. equates a single transposition to the flip of a single bit in the bitwise encoding that they devised to represent ordinal preferences in an election. The representation is a rather contrived construct designed to provide a simple plausible explanation for even trivial amounts of noise to have a real impact on the election outcome. Rather than delve into the semantics of representation, we will waive this and suffice it to say that some noise simply results in a specific type of alteration which is more significant than a single transposition. As with Procaccia et al., the forms of noise are intended to be unintentional and uncontrolled corruption, but some of the expanded noise may also have applicability to malicious manipulation of the system.

Procaccia et al. [27] is, in some ways, similar to Kalai's [22] work on noise sensitivity in social welfare functions. Kalai began with an assumption that votes are distributed uniformly at random, and investigated whether changes in a small percentage of voters' preferences can result in social preferences that differ from the originals. As mentioned above, there is also a large body of work dedicated to exploring the intentional manipulation of elections. Gibbard [16, 17] studied manipulation of elections in 1973 and 1977, and in 1990, Bartholdi, Tovey, and Trick [4] explored the difficulty of controlling an election. More recently, in 2003 and 2005, Conitzer, Sandholm, and Lang [9, 10] researched the computational complexity of manipulation. In the past few years, Faliszewski, Hemaspaandra, Hemaspaandra, and Rothe [13–15] have extensively examined the notions of bribery and control in elections, and the associated computational complexity. Our work is based heavily on the ideas of Procaccia et al. in that we explore the unintentional disruption of more prominent voting rules, with the goal of further exploring robustness

with increased noise.

The rest of the paper is organized as follows: Chapter 2 covers the history of elections, voting, and related research, including definitions of the voting rules we have studied; Chapter 3 provides an overview of our goals as well as the results of our research in the form of proofs of bounds on robustness; Chapter 4 describes in detail the experiments we performed and the resulting data; finally, Chapter 5 contains the conclusion, featuring a discussion of our results and the experimental data obtained. Our initial research concerning a form of noise we later chose not to focus on, called a suppression, can be found in Appendix A, and the source code for the program used to obtain experimental results is located in Appendix B.

Chapter 2

Background

2.1 History of Social Choice

Most historians point to Athens, the ancient Greek city-state of circa 500 BCE, as the birthplace of modern democracy. Though it is not generally considered to be the first democratic state, it is noted to be the most stable and important of the ancient world. Its form of government became a model not only for other Greek cities, but remains an important model today. The word democracy itself is in fact Greek, meaning power or rule (kratos) by the people (demos). The fundamental principle of Athenian democracy was that it was *aggregative*, providing the citizens an opportunity to influence the laws by which they are bound; this was in sharp contrast to the monarchies and oligarchies that preceded it. This basic tenet remains the foundation of modern democracies.

Although several of the world's foremost and famous mathematicians originated from ancient Greece, it does not appear that voting systems were studied mathematically until the 13th century, by Ramon Llull of Majorca [18]. Even these contributions were unknown until the rediscovery of Llull's manuscripts in the mid-to-late 20th century. A few of the most widely-known principles of election theory were discovered by Llull but lost over time, only to be independently rediscovered centuries later. One of these is the Borda voting system [26], a sequential descending scoring rule

made famous by Jean-Charles de Borda in 1770, which we study in depth in this paper; another is the Condorcet criterion [26], described by the Marquis de Condorcet in 1785, which requires the winner of an election to be preferred over every other candidate. We do not study the Condorcet criterion specifically, but two of the voting systems we do study, Copeland [11] and Maximin [25, 30], are Condorcet-compliant.

After the rediscovery of election theory by Condorcet and Borda in the late 1700s, new work continued through the next century, most notably by Charles Dodgson and Thomas Hare. Dodgson, more commonly known by his pen name Lewis Carroll as the author of *Alice's Adventures in Wonderland*, was also a mathematician and election theorist. Proposed in an 1876 pamphlet, the system now known as Dodgson's method is based upon the Condorcet criterion [26]. If there is no Condorcet winner, Dodgson proposed a method of determining a winner by choosing the candidate that is "closest" to meeting the criterion [26, 28].

Thomas Hare was a British lawyer and major proponent of electoral reform. Hare published several editions of his electoral theory work between 1857 and 1873, in which he created the Single Transferable Vote (STV) system still used in the Republic of Ireland and Australia [6], as well as the eponymous Hare quota that is sometimes used with STV. Hare, along with fellow election reform proponent and Member of Parliament John Stuart Mill, also popularized the idea of proportional representation, which is now used in parliamentary elections of many European countries (though ironically not in his home country).

Though many of the foundations of voting theory were laid in the works of Llull, Borda, and Condorcet, the bulk of work has been conducted in the 20th century and beyond. Several of the voting systems we will study were conceived within the last century, including Copeland (1951) [11], Maximin, also known as Simpson–Kramer [25, 30] (1969, 1977) or Minimax, and Bucklin [7] (1911). Other widely-known modern election systems include Black [5], Coombs [8] and Kemeny–Young [23, 24, 31].

Social choice theory, in its modern incarnation, was created by American economist Kenneth Arrow and popularized in his 1951 book, *Social Choice and Individual Values* [1]. Arrow's social choice theory is a blend of voting theory and welfare economics, essentially incorporating principles of sociology and economics to broaden the scope of voting theory. Arrow describes

how social values can be imposed by a "set of individual orderings," essentially a preference profile consisting of ranked votes, aggregated under a "constitution," which is a voting rule that maps a set of orderings to one "social ordering."

An important component of Arrow's book is a theorem now commonly known as Arrow's impossibility theorem, or Arrow's paradox. Arrow decreed four "reasonable" requirements of any voting system—unanimity (also called Pareto efficiency), unrestricted domain (also called universality), nondictatorship, and independence of irrelevant alternatives—and mathematically proves that no voting system allowing more than two choices can ever uphold all of these principles simultaneously. The theorem is sometimes (controversially) condensed to statements such as "No voting method is fair." These principles as written by Arrow quickly became an important framework for studying social choice that is still in place today.

In the past two decades, a new discipline has arisen known as computational social choice, in which the studies and principles of computer science are applied to problems of social choice theory. The seminal work in this area is generally considered to be a 1989 article by Bartholdi, Tovey, and Trick [3]. This work provides a proof that in a Dodgson election, in terms of computational complexity it is NP-hard to simply determine if a particular candidate is a winner of the election. Perhaps more importantly, the work also provides a class of "impracticality theorems" which are somewhat analogous to Arrow's impossibility theorem, as they assert that any fair voting scheme must require excessive computation to determine a winner in the worst case.

Just a few months after their first article, Bartholdi, Tovey, and Trick published a second article [2] that specifically addresses computational complexity of manipulating an election, as well as an election system that is resistant to computational manipulation. This work provided much of the foundation for future study of computational complexity. Hemaspaandra, Hemaspaandra, and Rothe have written extensively on this subject in the past decade, covering topics such as the complexity of Dodgson elections [19], manipulating elections to prevent a specific candidate from winning [21], and resisting manipulation by using hybrid elections [20].

Conitzer and Sandholm [10] studied the manipulability of several voting rules in the context of multiagent systems. By bounding the elections in

such a way that there are relatively few candidates, the work is able to provide specific bounds on the computational complexity of manipulating several different election rules. Both individual manipulation and "coalitional" manipulation by several agents in concert were considered. Conitzer and Sandholm were able to determine that given information about other agents' votes, manipulation is very easy, even by an individual agent, as long as votes are unweighted. They also concluded that manipulation under a system of weighted votes is generally intractable. The conclusions of this work and related works are generally that manipulation of an election is easy, especially if the intended manipulation is "destructive," meaning the goal is simply to prevent one candidate from winning.

The emerging field of computational social choice was formalized in December 2006 with its first international workshop [12]. It is from the proceedings of this conference that the primary basis for our research comes: a study on the robustness of election systems by Procaccia et al. [27]. The work is certainly not the first research pertaining to this field, however. Kalai [22] essentially studied robustness of elections, but without using the specific term "robustness." Kalai's work proposed the more general question of "How likely is it that small random mistakes in counting the votes in an election between two candidates will reverse the election's outcome?" Assuming a uniform and independent distribution of preferences, the work uses social welfare functions with simple voting rules to analyze the robustness. Kalai also defined the notion of "social chaos," which is related to the probability of finding cycles in the preferences. Interestingly, Kalai was also able to relate the the robustness under random noise the likelihood that a candidate a is preferred to c, given that a is preferred to b and b is preferred to c. As a whole, Kalai's work is concerned more with social welfare functions and random noise, rather than the specific voting rules and noise types of our research.

Computational social choice has recently gained the attention of artificial intelligence researchers who seek to use voting methods to conduct preference aggregation in multiagent systems. Since such preference aggregation often occurs in networked computational environments, security concerns become a major factor. Rather than random noise, these systems are susceptible to manipulation, from both external entities who have acquired access to the network, and especially by AI components of the multiagent system, which may seek to manipulation the election they are participating in. By altering their votes in such a manner as to not represent their actual preferences,

but rather to affect the outcome of the election in some way.

The robustness of an election is defined to be the probability for which an election's outcome will remain unchanged given a certain disruption in the voting data. Procaccia et al. [27] investigates the robustness of various common voting systems with respect to *elementary transpositions*. An elementary transposition is essentially the least significant possible change that can be made to an election: in a single voter's ranked preference list of candidates, the positions of two adjacent candidates are swapped. The work focuses specifically on the "1-robustness," which is the probability of robustness given a single such transposition. Procaccia et al. provides upper and/or lower bounds on the 1-robustness of five voting systems: scoring rules, Copeland, Maximin, Bucklin, and Plurality with Runoff.

2.2 Notation

Procaccia et al. [27] uses conventional social choice notation from Brams and Fishburn [6] to mathematically represent elections and their components. In order to assist in both viewing this work as an extension of Procaccia et al., as well as within the context of computational social choice, we will follow this same basic notation, with a few of our own additions.

Let the set of voters be $V = \{v^1, v^2, \ldots, v^n\}$ and the set of candidates be $C = \{c_1, c_2, \ldots, c_m\}$ where $n = |V|$ and $m = |C|$. The index i in superscript refers to voters, and the index j in subscript refers to candidates. The set of all linear orders on C is denoted by $\mathcal{L} = \mathcal{L}(C)$. Each voter i has ordinal preferences $\succ^i \in \mathcal{L}$ where the candidates are ranked $c_{j_1} \succ^i c_{j_2} \succ^i \cdots \succ^i c_{j_m}$. $\succ^V = \langle \succ^1, \ldots, \succ^n \rangle \in \mathcal{L}^N$ is a *preference profile*. $\pi_l(\succ^i)$ denotes the candidate that voter i ranks in the l'th position; similarly, the notation l^i_j indicates the ranking of candidate c_j in ordinal preference list $\succ^i$. The winner of an election is decided by the voting rule, which is a function $F : \mathcal{L}^V \to C$. This mapping of preferences to candidates designates the winning candidate.

2.3 Definitions of Voting Rules

The voting rules we have chosen to focus on are scoring rules, Copeland, Maximin, Bucklin, and plurality with runoff. It is intentional that these are the same rules studied in Procaccia et al. [27], as this allows us to make a direct comparison between the results on robustness of elementary transposition and arbitrary reordering. These rules are all firmly established and well-studied within the field of social choice. This section defines, both formally and informally, each of these rules. Though there is no one universal method for tie-breaking an election in which more than one candidate has the highest score, for the purposes of this work, a winning candidate will be selected arbitrarily from the set of candidates with the highest score.

2.3.1 Scoring Rules

Scoring rules are actually representative of a generic class of election rules, rather than a specific rule; however, since they can all be generically represented in the same way, we can study them together. Scoring rules are based upon a scoring vector $\vec{\alpha}$ of size m that provides a score for each rank. A candidate's score in a given preference profile is simply the sum of its scores based on its rank in each vote. The formal definition is as follows: given a scoring vector $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_m \rangle$, the score of a candidate j is $s_j = \sum_i \alpha_{l_j^i}$, and the winner of the election is $F(\succ) = \text{argmax}_j s_j$.

Though the scoring vector $\vec{\alpha}$ is not constrained to any specific values, there are some common scoring rules that have been studied (we were unable to find authoritative references for plurality and veto, but these definitions are widely accepted). Although our results are general and applicable to all scoring rule implementations, we will focus these specific scoring vectors for our experimental results:

- Borda: $\vec{\alpha} = \langle m-1, m-2, \ldots, 0 \rangle$ [26]
- Plurality: $\vec{\alpha} = \langle 1, 0, \ldots, 0 \rangle$
- Veto: $\vec{\alpha} = \langle 1, \ldots, 1, 0 \rangle$

2.3.2 Copeland [11, 27]

The Copeland election rule is based upon the rankings of each candidate relative to every other candidate. A series of pairwise elections is conducted, each of which compares exactly two candidates to each other. The resulting scores of the pairwise election are simply the number of times each candidate was ranked higher than the other; the degree by which their ranks differ is not a factor. For Copeland, every candidate competes in a pairwise election against ever other candidate, and each candidate's score is the number of other candidates they beat. In the original Copeland definition, ties were awarded half points; in some later works, including Procaccia et al. [27], this is not done. The formal definition of Copeland as used in Procaccia et al., and for the purposes of this work, is as follows:

Candidate j beats j' in a pairwise election if $|\{i : l_j^i < l_{j'}^i\}| > n/2$. The score for candidate j, s_j, is the number of candidates that j beats in pairwise elections. The winner of the election Copeland($\succ$) is $\text{argmax}_j s_j$.

2.3.3 Maximin [25, 30]

Maximin, sometimes called Minimax or Simpson–Kramer, consists of a series of pairwise elections comparing each candidate to every other candidate, similar to Copeland. In Maximin, however, the candidate's score is its *worst* pairwise score against the other candidates. The winner of the election is still the candidate with the highest score, however, which could be thought of as "the best of the worst." Formally, the definition of Maximin is:

The score of candidate j is $s_j = \min_{j'} |\{i : l_j^i < l_{j'}^i\}|$, and the winner of Maximin($\succ$) is $\text{argmax}_j s_j$.

2.3.4 Bucklin [7]

The Bucklin election system, while somewhat confusing in its mathematical definition, is fairly simple in principle. A candidate needs a majority score, more than half of n, to win. The election proceeds in rounds. The first round looks only at the first-ranked candidate in each vote. If no candidate

has a majority, the election proceeds to the second round, where the top two ranks are considered, and so forth, iterating through each successive rank. A candidate's score is simply the total number of combined votes it has in all currently-considered ranks. If at any time a candidate reaches a majority score, the election is stopped, and that candidate is declared the winner. Formally, Bucklin is defined as:

For all candidates c_j and $l \in \{1, \ldots, m\}$, let $B_{j,l} = \{i : l_j^i \leq l\}$. The winner of Bucklin($\succ$) is $\text{argmin}_j(\min\{l : |B_{j,l}| > n/2\})$.

2.3.5 Plurality with Runoff[9]

Plurality with Runoff is a hybrid election, which always consists of two rounds. The first round is a plurality election, as defined above in Scoring Rules; however, rather than a single winner, this first round returns the two candidates with the highest plurality scores. These two candidates proceed to a runoff election, which is pairwise. The runoff candidate with the higher pairwise score wins the overall election.

More formally, the two candidates who maximize $|\{i \in N : l_j^i = 1\}|$, move on to a pairwise runoff election for the second round, of which the winner is the candidate j such that $|\{i : l_j^i < l_{j'}^i\}| > n/2$.

Chapter 3

Robustness in Voting

Function	Lower Bound		Upper Bound	
	Transposition	Reordering	Transposition	Reordering
Scoring Rule	$\frac{m-1-a_F}{m-1}$	$\frac{(\lfloor \frac{m}{a_F+1} \rfloor !)^{a_F+1}}{m!}$	$\frac{m-a_F}{m}$	$1 - \frac{2m(a_F-1)-{a_F}^2-a_F+1}{2m!}$
Copeland	0^*	$0^\dagger$	$\frac{1}{m-1}$	$1 - \frac{(m+1)(m-1)!-2}{2m!}$
Maximin	0^*	$0^\dagger$	$\frac{1}{m-1}$	$1 - \frac{(m-1)!-1}{m!}$
Bucklin	$\frac{m-2}{m-1}$	$\frac{\left(\left(\frac{m-1}{2}\right)!\right)^2}{m!}$	1^*	$1^\dagger$
Plurality w. Runoff	$\frac{m-5/2}{m-1}$	$\frac{7}{6m}^\ddagger$	$\frac{m-5/2}{m-1} + \frac{5/2}{m(m-1)}$	$\frac{9m-10}{4m^2}^\ddagger$

Table 3.1: Comparison of the upper and lower bounds of various scoring rules for elementary transposition and arbitrary reordering. The bounds for elementary transposition are from Procaccia et al. [27]; the bounds for arbitrary reordering are from our own work and discussed in detail in this chapter.

*These bounds are as provided in [27], but it is unlikely they represent actual bounds. In these cases the authors indicated they felt the actual bounds were inconsequential.

†These values are not intended to denote the actual bounds, but simply indicate these particular values were beyond the scope of this work and have been replaced by the absolute minimum or maximum bound.

‡These bounds only hold for values of $m \geq 3$, but we find this to be a reasonable assumption on the minimum size of an election.

3.1 Overview

Recently, Procaccia et al. [27] conducted research on robustness in voting. Specifically, the work was focused on the the smallest possible amount of noise, the single elementary transposition. Procaccia et al. derived upper and/or lower bounds for five different established voting rules. The authors offer a specific binary representation of a preference profile in order to supplement their fault model. Specifically, the representation uses a single bit to represent each of the $\binom{m}{2}$ candidate pairs; a value of 1 indicates the first candidate is preferred to the second, and a value of 0 is a preference for the second candidate over the first. While it is intuitive that this is not the most compact form of representation, which the authors note, they do elaborate on some advantages; specifically, there is the possibility to check for certain properties of the election in constant time using bitmasks. It has a major disadvantage in that it is possible to represent an inconsistent preference profile, in that there the transitivity can be violated. While this is not difficult to detect, it is still an impossible occurrence in a more conventional list-based representation.

We will expand the notion of noise to include more significant alterations to the preference profile; specifically, we will examine the impact of a voter's ordinal preferences being scrambled (arbitrarily reordered), which itself is actually equivalent to a series of elementary transpositions confined to the context of a single voter. In our preliminary research, we also investigated the notion of suppression, in which a single voter's entire ordinal preference list is not considered in determining the election outcome; however, we were unable to obtain interesting results from this, and consequently removed it from the final work. The preliminary work regarding suppression can be found in Appendix A.

While the Procaccia et al. [27] representation certainly provides a justification for allowing the smallest possible data corruption, a single but flip, to impact the outcome of the election, we did not find it necessary in our work to focus on a specific representation of our fault model. The pairwise binary representation is somewhat contrived in that its design is admittedly inefficient and exists largely to bolster the contention that even minor data corruption can impact an election. While it may be worthwhile to provide a concrete example of the potential ramifications of such a corruption, we do not feel it necessary to expound upon this any further. We find the imple-

mentation details to be generally irrelevant to the actual impact of the noise on the robustness of the election. Given these expanded criteria for noise, we will examine the effects on the k-robustness of the same election rules as Procaccia et al. [27]: basic scoring rules (Plurality, Veto, and Borda), Copeland, Maximin, Bucklin, and Plurality with Runoff.

3.2 Definitions

Procaccia et al. [27] focuses exclusively on one type of noise, the *elementary transposition*. This consists of a swap between the rankings of two adjacent candidates in a single voter's ordinal preference list, and is arguably the "smallest" amount of noise possible. As such, it can even be considered the building block of more exaggerated forms of noise, and we will use it as such. We include here the Procaccia et al. definition of elementary transposition:

Definition 1. A preference profile $\succ_1^V$ is obtained from a preference profile $\succ^V$ by an *elementary transposition* (write: $\succ^V \rightsquigarrow \succ_1^V$) if there exists a voter v^i and $l \in \{2, \ldots, m\}$ such that:

1. for all $i' \neq i$, $\succ^{i'} = \succ_1^{i'}$
2. $\pi_l(\succ^i) = c = \pi_{l-1}(\succ_1^i)$
3. $\pi_{l-1}(\succ^i) = c' = \pi_l(\succ_1^i)$
4. $\succ^i \downarrow_{C \setminus \{c,c'\}} = \succ_1^i \downarrow_{C \setminus \{c,c'\}}$

The $\succ^i \downarrow_{C \setminus \{c,c'\}}$ notation is used by Procaccia et al. [27] without a specific definition, but we understand it to be defined as the preference profile $\succ^i$ with candidates $\{c, c'\}$ removed from consideration.

We build upon this concept of a single transposition to create a more severe form of noise, which we call an *arbitrary reordering*. The reordering consists of a random permutation of a single voter's ordinal preference list. This form of noise is strongly linked to elementary transpositions, as an arbitrary reordering could easily be defined to consist of a series of multiple elementary transpositions confined to a single voter within the preference profile. Arbitrary reordering is formally defined as follows:

Definition 2. Given a preference profile $\succ^V$ and a voter $v^i \in V$, the preference profile $\succ_1^V$ is obtained from $\succ^V$ by an *arbitrary reordering* if for all $i' \neq i$, $\succ^{i'} = \succ_1^{i'}$.

We measure the effects of noise on an election or voting rule in terms of robustness: the probability that given some sort of noise, disruption, or alteration of the preference profile data, the outcome of the election is unaffected. Specifically, we are interested in the *k-robustness*, which is defined here as the probability of robustness given k independent noise faults:

Definition 3. The *k-robustness* of a preference profile $\succ^V$ under an arbitrary reordering of k voters $v^i \in V$ for $i = 1, \ldots, k$, written $\phi_k(\succ^V) = \succ_1^V$, is:

$$\rho_{\phi_k}(F, \succ^V) = \Pr_{\succ_1^V \sim \phi_k(\succ^V)} \left[F(\succ^V) = F(\succ_1^V) \right].$$

The notation $\succ_1^V \sim \phi_k(\succ^V)$ is adapted from Procaccia et al. [27], in which it is defined to be the probability of obtaining $\succ_1^V$ from $\succ^V$ by k faults chosen randomly and independently.

Once the k-robustness of the preference profile is established, it is straightforward to link the robustness to the voting rule itself:

Definition 4. The *k-robustness* of a voting rule F under arbitrary reordering of k voters $v^i \in V$ for $i = 1, \ldots, k$ with n voters and m candidates is:

$$\rho_{\phi_k}^{n,m}(F) = \min_{\succ_1^V \in \mathcal{L}(C)^n} \rho_{\phi_k}(F, \succ_1^V).$$

3.3 Theorems

Largely in order to simplify our contention that the arbitrary reordering is strongly linked to the elementary transposition, which consequently simplifies proofs of our work in relation to Procaccia et al. [27], we explicitly prove the relation:

Proposition 1. *Every arbitrary reordering can be obtained by a series of at most* $\frac{m^2-m}{2}$ *elementary transpositions.*

Proof. Since the arbitrary reordering is confined to a single voter's ordinal preference list, $\succ^i$, the elementary transpositions take place on one list of size m, i.e. $\mid \succ^i \mid = m$. It therefore can take at most $m-1$ transpositions to get the first list element in $\succ^i$ to match its position in the arbitrarily reordered list; that is, to order the list such that $\pi_1(\succ_1^i) = \pi_1(\succ^i)$. Subsequently, it will take at most $m-2$ transpositions for the second element, $m-3$ for the third, etc., resulting in the summation $\sum_{h=1}^{m-1} m-h$. It is straightforward to see that this summation is equal to $\frac{m^2-m}{2}$, which is the upper bound to obtain an arbitrary reordering via a series of elementary transpositions. □

Following the proof of Proposition 1, it is a simple case to prove that an arbitrary reordering can alter the outcome of an election. Procaccia et al. [27] already proves this for elementary transpositions in Theorem 1, included below.

Theorem 1 [27]. *Let $F : \mathcal{L}^V \to C$ be a voting rule such that Range$(F) > 1$. Then there exists a preference profile $\succ^V$ and a profile $\succ_1^V$ which is obtained from $\succ^V$ by an elementary transposition, such that $F(\succ^V) \neq F(\succ_1^V)$.*

Theorem 2. *Let $F : \mathcal{L}^V \to C$ be a voting rule such that Range$(F) > 1$. Then there exists a preference profile $\succ^V$ and a profile $\succ_1^V$ which is obtained from $\succ^V$ by an arbitrary reordering, such that $F(\succ^V) \neq F(\succ_1^V)$.*

Proof. Since $\succ_1^V$ can be obtained from $\succ^V$ by a series of elementary transpositions, per Proposition 1, the proof of this claim follows inherently from the Procaccia et al. [27] proof for Theorem 1. □

To provide a link between k-robustness and 1-robustness, Procaccia et al. [27] bounds k-robustness by the kth power of 1-robustness, as seen below in their Proposition 2. We do the same, but define k in terms of an arbitrary reordering.

Proposition 2 [27]. $\rho_k^{n,m}(F) \geq (\rho_1^{n,m}(F))^k$.

Proposition 3. *Given k arbitrary reorderings, $\rho_{\phi_k}^{n,m}(F) \geq (\rho_{\phi_1}^{n,m}(F))^k$.*

Proof. Consider the preference profile $\succ_1^V$ and the preference profile $\succ_2^{V^k}$ obtained from it by k independent and random arbitrary reorderings. We claim the probability that $F(\succ_1^V) = F(\succ_2^{V^k})$ is at least $(\rho_{\phi_1}^{n,m})^k$. Let $\succ_{i_1}^V, \ldots, \succ_{i_{k+1}}^{V^k}$ be the intermediate preference profiles obtained by the reorderings, where $\succ_{i_1}^V = \succ_1^V$, $\succ_{i_{k+1}}^{V^k} = \succ_2^V$, and each $\succ_{i_{j+1}}^V$ is obtained from $\succ_{i_j}^V$ by a random and independent arbitrary reordering for $j = 1, \ldots, k$. By the definition of 1-robustness, for every preference profile $\succ^{V^i}$, the probability that one randomly chosen arbitrary reordering does not change the outcome of the election under F is at least $\rho_{\phi_1}^{n,m}(F)$. Therefore, for $j = 1, \ldots, k$,

$$\Pr\left[F(\succ_{i_j}^V) = F(\succ_{i_{j+1}}^V) \mid \succ_{i_j}^V\right] \geq \rho_{\phi}^{n,m}(F).$$

Therefore:

$$\begin{aligned}\Pr\left[F(\succ_1^V) = F(\succ_2^V)\right] &= \prod_{j=1}^{k} \Pr\left[F(\succ_{i_j}^V) = F(\succ_{i_{j+1}}^V) \mid \succ_{i_j}^V\right] \\ &\geq (\rho_{\phi_1}^{n,m})^k.\end{aligned}$$

□

Above, we have used the concept of 1-robustness as a lower bound for the k-robustness. This is done in the same manner as Procaccia et al. [27] and for exactly the same reasons: a high lower bound implies a high k-robustness, while a low 1-robustness indicates that the k-robustness of the rule is not worth considering.

Now that we have clearly defined our models for noise, as well as the definitions of their robustness, we will apply these in order to examine the k-robustness of the voting rules (Plurality, Veto, Borda, Copeland, Maximin, and Plurality with Runoff) in much the same manner as Procaccia et al. [27], with the exception of the definition of a fault. Rather than each of the k faults indicating a single elementary transposition, we will redefine a fault within the context of our noise; that is, k instances of arbitrary reordering.

Recall that the specific type of noise we are working with, the "arbitrary reordering," is in effect simply a permutation of a single vote. In bounding the robustness of the election rules below, we may refer to the arbitrary reordering in a mathematical context as a permutation. For all of the rules

we discuss, there are m candidates, and therefore in each vote there are $m!$ possible permutations, or arbitrary reorderings. We may refer to a set of permutations as "safe" if no permutation of that form can possibly alter the election outcome, thereby contributing to the quantification of robustness.

3.4 Scoring Rules

In this section, we attempt to quantify the robustness generally for all scoring rules. It is difficult to obtain meaningful results for such a broad class of rules, as the scoring vector $\vec{\alpha}$ significantly affects the resulting robustness. Given a scoring rule F with scoring vector $\vec{\alpha}$, let $A_F = \{l \in \{2, \ldots, m\} : \alpha_{l-1} > \alpha_l\}$, and $a_F = |A_F|$. The robustness of a scoring rule is strongly linked to the parameter a_F, as the scores can only change when the noise violates the boundaries in the scoring vector. To find the lower bound for scoring rules, we claim that the worst case distribution is when each of the score groups contained within in A_F are of equal size.

Proposition 1. *Let n and m be the number of voters and candidates, and let F be a scoring rule. Then $\rho_1^{n,m} \geq \dfrac{(\lfloor \frac{m}{a_F+1} \rfloor !)^{a_F+1}}{m!}$.*

Proof. When there are a_F divisions in the scoring vector, there are $a_F + 1$ distinct groups of equal scores. One way to guarantee that the outcome of the election remains the same after a vote is reordered is to require that the reordering causes no candidate to "jump" across the divisions in A_F that bound each score group. For each of the $a_F + 1$ groups of size r_s, there are $r_s!$ possible permutations that meet this restriction. Therefore, the total number of acceptable permutations is $\prod_{s=1}^{a_F+1} r_s!$.

This value must be minimized when the score groups in $\vec{\alpha}$ are as close to equal as possible; that is, when the size of each group is between $\lfloor \frac{m}{a_F+1} \rfloor$ and $\lceil \frac{m}{a_F+1} \rceil$. When this is the case, the probability that the outcome of the election F will not change must be at least $\dfrac{(\lfloor \frac{m}{a_F+1} \rfloor !)^{a_F+1}}{m!}$. □

The upper bound is similarly dependent on the score groupings in A_F. In

this case, we find that skewing the distribution to one end, just the opposite of the equal groupings used in the lower bound, allows us to provide a tighter upper bound.

Proposition 2. *Let n and m be the number of voters and candidates, and let F be a scoring rule. Then* $\rho_1^{n,m} \leq 1 - \dfrac{2m(a_F - 1) - {a_F}^2 - a_F + 1}{2m!}$.

Proof. Given that there are $a_F + 1$ groups of equal scores in the scoring vector $\vec{\alpha}$, we can guarantee a change in the outcome of the election F by strictly increasing the score of one candidate while strictly decreasing the score of another. We can accomplish this by reordering a vote such that two candidates associated with different score groups switch places in the ordinal preferences, as long as neither of the candidates is the winning candidate c^W.

For each candidate $c_j \in C$, we can swap with any of the other m candidates, less the candidates in c_j's own scoring group and the winning candidate c^W. Denoting the size of candidate c_j's scoring group as a_F^j, the number of such swaps is the sum $\sum_{j \in C - \{c^W\}} m - a_F^j - 1$. Removing the constant terms yields the sum $(m-1)^2 - \sum_{j \in C - \{c^W\}} a_F^j$. In order to maximize the number of outcome-altering reorderings, a_F of the score groups in $\vec{\alpha}$ must be of size 1, and one such group must contain the winning candidate. The remaining score group is of size $(m - a_F)$. Since $a_F^j = 1$ for all but one score group, the expanded summation is $(m-1)^2 - a_F - (m - a_F)^2$. To correct double counting, this entire value is divided by 2. Subtracting from 1 to find the robustness, the result is $1 - \dfrac{2m(a_F - 1) - {a_F}^2 - a_F + 1}{2m!}$. □

3.5 Copeland

To provide an upper bound on Copeland, we use a vote distribution in which all votes are divided into two groups with inverted ordinal preferences. Although it may seem at first glance that the rank of the winning candidate will affect the number of safe permutations, we prove that because each candidate has a corresponding rank in the second inverted vote group, the

rank of the candidate within any randomly selected vote is irrelevant in considering the robustness of the rule.

Proposition 3. *Let m be the number of candidates, and let n, the number of voters, be even. Then* $\rho_1^{n,m}(Copeland) \leq 1 - \dfrac{(m+1)(m-1)! - 2}{2m!}$.

Proof. Consider an election in which for $i = 1, 3, 5, \ldots, n-1$, the ordinal preferences of voters v^i and v^{i+1} fall into one of two preference profiles:

$\succ^i$	$\succ^{i+1}$
c_1	c_m
c_2	c_{m-1}
.	.
.	.
c_m	c_1

In the above scenario, for every two candidates c and c', exactly $n/2$ voters prefer c. Therefore, the Copeland score for every candidate is 0, and the winner is an arbitrary candidate $c \in C$. Given the winning candidate c_j, where j is the candidate's rank in $\succ^i$, we can guarantee the outcome of the election will change under an arbitrary reordering in which the ranking of c_j strictly decreases in any one vote, or in which the rank of any other candidate increases.

If such a vote is contained by the preference profile $\succ^i$, there are $m - j$ decreasing positions in which to place c_j, and therefore $(m - j)(m - 1)!$ permutations which decrease c_j's score. It is also possible to alter the outcome of the election by increasing the score of any other candidate, which is achieved by fixing the position of c_j and allowing any other permutation. Therefore, subtracting 1 to remove the identity permutation, there are $(m-j+1)(m-1)!-1$ permutations guaranteed to alter the outcome of the election. The alternative, with equal probability, is that the altered vote is contained in the preference profile $\succ^{i+1}$. In this case, there are $j-1$ decreasing positions in which to place c_j, and therefore $j(m-1)!-1$ permutations guaranteed to alter the outcome of the election.

The probability of reordering a vote such that the outcome of the election will change is then $\frac{1}{2}\left[(m-j+1)(m-1)!-1\right]+\frac{1}{2}\left[j(m-1)!-1\right]$. Factoring out the common $\frac{1}{2}(m-1)!$, it is clear that the j terms cancel and thus the position j of the winning candidate c_j is irrelevant in calculating the num-

ber of outcome-altering reorderings. Taking into account the total number of permutations, $m!$, the probability of changing the election outcome is $\frac{(m+1)(m-1)!-2}{2m!}$. Subtracting from 1 to find the robustness, the result is $\rho_1^{n,m}(Copeland) \leq 1 - \frac{(m+1)(m-1)!-2}{2m!}$. □

3.6 Maximin

To find an upper bound for the robustness of Maximin, we construct an adversarial preference profile with specific properties that ensures each candidate's score is tied. This distribution ensures the promotion of any candidate within any preference profile, except for the winner, changes the outcome of the election. By maximizing the probability a reordering will promote a non-winning candidate, we thereby minimize the upper bound. Our proof uses from Proposition 8 of Procaccia et al. [27], which is included below.

Proposition 8 [27]. *Let n and m be the number of voters and candidates such that m divides n. Then $\rho_1^{n,m}(Maximin) \leq 1/(m-1)$.*

Proposition 4. *Let m be the number of candidates and n be the number of voters such that m divides n. Then $\rho_1^{n,m}(Maximin) \leq 1 - \frac{(m-1)!-1}{m!}$.*

Proof. We will iteratively construct the preference profile in the same manner as Procaccia et al. does in Proposition 8. In the first iteration, each voter's only linear preference is candidate c_1. In the next iteration, candidate c_2 is ranked first for $\frac{1}{m}n$ voters, and below c_1 for the remaining $\frac{m-1}{m}n$ voters. In each subsequent iteration, candidate c_j is ranked first among the $\frac{1}{m}n$ candidates who ranked c_{j-1} last; the remaining candidates rank c_j immediately below c_{j-1}. For example, given 8 voters and 4 candidates, the resultant preference profiles would be as follows:

$\succ^1$	$\succ^2$	$\succ^3$	$\succ^4$	$\succ^5$	$\succ^6$	$\succ^7$	$\succ^8$
c_2	c_2	c_3	c_3	c_4	c_4	c_1	c_1
c_3	c_3	c_4	c_4	c_1	c_1	c_2	c_2
c_4	c_4	c_1	c_1	c_2	c_2	c_3	c_3
c_1	c_1	c_2	c_2	c_3	c_3	c_4	c_4

Each candidate c_j is ranked in each position $j \in \{1, 2, \ldots, m\}$ exactly $\frac{n}{m}$ times. Given a winning candidate c^w contained within any preference profile, we can guarantee a change in the outcome of the election under an arbitrary reordering by fixing the position of c^w and reordering the remaining $m-1$ candidates in the profile. This yields $(m-1)!$ reorderings that alter the election outcome, less one for the identity permutation. Considering the total possible $m!$ permutations and subtracting from one to find the robustness, the result is $\rho_1^{n,m}(Maximin) \leq 1 - \dfrac{(m-1)!-1}{m!}$. □

3.7 Bucklin

The Bucklin rule progressively considers each successive ranking until a plurality is reached. We define the threshold rank at which this occurs as l_0. By proving permutations isolated by the threshold as a barrier, we also prove lower bound by maximizing the number of safe permutations.

Proposition 5. *Let m be the number of candidates and n be the number of voters. Then $\rho_1^{n,m}(Bucklin) \geq \dfrac{\left(\left(\frac{m-1}{2}\right)!\right)^2}{m!}$.*

Proof. Assume candidate c_j is the winner of the election and satisfies $l_0 = \min_l B(j,l) > n/2$; in other words, l_0 is the lowest rank in the preference profile considered when determining the winner of the election. We claim that under an arbitrary reordering, the outcome of the election does not change as long as the candidate ranked in position l_0 is fixed and the candidates ranked above and below l_0 do not cross the "boundary" of l_0. There are three cases to consider for proof of this claim:

Case 1: For any voter i and any candidate c_k where $l_k^i < l_0$, consider any promotion or demotion c_k such that $l_k^i < l_0$ still holds true. No such change in rank can alter the outcome of the election, as $B(k, l_1)$ remains unchanged for all $l_1 \leq l_0$.

Case 2: For any voter i and any candidate c_k where $l_k^i > l_0$, any change of rank of c_k such that $l_k^i > l_0$ still holds true cannot alter the outcome of the election. The condition $\min_l B(j,l) > n/2$ is already satisfied at position

l_0; therefore, no candidate whose rank $l_k^i > l_0$ could affect the score or the election outcome.

Case 3: For any voter i and any candidate c_k where $l_k^i < l_0$, it is easy to see that a demotion with resulting rank $l_k^i > l_0$ can alter the election outcome if $c_k = c_j$. Likewise, a promotion of a candidate c_k where $l_k^i > l_0$ with resulting rank $l_k^i < l_0$ could potentially alter the outcome of the election if $c_k \neq c_j$.

The fourth case not explicated is a change of rank for candidate c_k with rank $l_k^i = l_0$. It is straightforward to see that changes at position l_0 may alter the outcome of the election.

For any ordinal preference list $\succ^i$, fixing the position of the candidate $\pi_{l_0}(\succ^i)$ results in two "groups" of candidates that may be reordered per the above cases. For each group of size r_s, the number of permutations which cannot affect the election outcome is $r_s!$. The worst case is therefore the one that minimizes the size of the two groups, which occurs if $l_0 = m/2$, and the size of each group is $\frac{m-1}{2}$. Therefore, the robustness is $\rho_1^{n,m}(Bucklin) \geq \frac{\left(\left(\frac{m-1}{2}\right)!\right)^2}{m!}$. □

3.8 Plurality with Runoff

Plurality with runoff presents a unique challenge in that it utilizes two different election rules in two rounds; therefore, the effect of arbitrarily reordering a vote must be considered for both rounds. We construct an adversarial preference profile similar to that which we used for Maximin, in that all m candidates have equal scores in the plurality election. In order to derive the worst case distribution, we maximize the potential for altering the election outcome by ranking the arbitrary winners of the plurality election in last place for all remaining votes. In such a distribution, the promotion of any candidate to first place will affect the election result.

Proposition 6. *Let n be the number of voters and m be the number of candidates, such that $m \mid n$. Then for all $m \geq 3$, $\rho_1^{n,m}$(Plurality with Runoff) $\leq \frac{9m-10}{4m^2}$.*

Proof. Our preference profile is constructed as follows: The n votes are distributed into m groups of identical votes, each of equal size $\frac{n}{m}$. Each group ranks its eponymous candidate first, e.g. $A = \{\succ^i\colon \pi_1(\succ^i) = c_a\}$. In groups A and B, the rank of the remaining candidates is irrelevant. For the remaining groups, candidates c_a and c_b are ranked in the last two positions: half such that $\pi_m(\succ^i) = c_a$, and half such that $\pi_m(\succ^i) = c_b$; the order of the remaining candidates is again irrelevant. For display purposes, the irrelevant candidates are ranked in order. The resulting distribution is the following:

A	B	C	D	$\cdots$
c_a	c_b	c_c	c_d	
c_b	c_a	c_d	c_c	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
c_y	c_y	c_a	c_b	
c_z	c_z	c_b	c_a	

In this distribution, all candidates have equal scores in the plurality election, and candidates c_a and c_b are chosen as the winners to compete in the pairwise runoff. In the runoff election, candidates c_a and c_b have equal pairwise scores, and c_a is chosen as the winner of the election.

For votes in group A, demotion of c_a (and thus promotion of some other candidate to first place) will reduce c_a's score such that it may no longer be a winner of the plurality election. Votes in group B are similar in that demoting c_b reduces that candidate's score such that it may no longer be a winner of the plurality election, and any replacement candidate has a higher pairwise score than c_a. Therefore, the first-ranked candidates in groups A and B must remain fixed to ensure there is no change in the election outcome.

The remaining vote groups are all identical in terms of the effect of an arbitrary reordering: the first-ranked candidate may not be demoted and replaced by another candidate except for c_a or c_b, as the promotion of other candidates to first place will guarantee that candidate a win in both the plurality and the runoff elections. Even when the top candidate is fixed, however, exactly half of the votes rank c_b below c_a, and half the permutations on the lower candidates will alter the relative positions of c_a and c_b; therefore, there is a $\frac{1}{4}$ probability a permutation of the bottom $m-1$ candidates will

increase c_b's pairwise score over c_a, and thus change the outcome of the election. When c_a is promoted to first place, all permutations of the lower $m-1$ candidates are safe, as c_b will always be ranked lower. If c_b is promoted to first place, c_b will clearly rank above c_a for all permutations of the lower candidates, but c_b's pairwise score relative to c_a only increases for the half of the votes in which c_b was ranked below c_a to begin with.

Since there are m groups, the probability of choosing either group A or B is $\frac{2}{m}$. The probability a permutation does not demote the first place candidate is $\frac{(m-1)!}{m!}$. The probability of choosing any other groups is $\frac{m-2}{m}$. For these groups, the probability a permutation is safe is $\frac{3}{4}$ if the existing winner is fixed, 1 if c_a is promoted to first, and $\frac{1}{2}$ if c_b is promoted to first, as described above. The total probability of a safe permutation is therefore $\frac{3(m-1)!}{4m!} + \frac{(m-1)!}{m!} + \frac{(m-1)!}{2m!} = \frac{9(m-1)!}{4m!}$. The resulting probability of robustness is $\frac{2}{m} \cdot \frac{(m-1)!}{m!} + \frac{m-2}{m} \cdot \frac{9(m-1)!}{4m!} = \frac{9m-10}{4m^2}$. □

Proposition 7. *Let m be the number of candidates and n be the number of voters. Then for all $m \geq 3$, $\rho_1^{n,m}$(Plurality with Runoff) $\geq \frac{7}{6m}$.*

Proof. Given candidate c_a as the winner of the overall election, by definition c_a is also one of two winners of the plurality election. There must then be a second winner of the plurality election, candidate c_b. We define set V_a to be $\{\succ^i : \pi_1(\succ^i) = c_a\}$, V_b to be $\{\succ^i : \pi_1(\succ^i) = c_b\}$, $n_a = |V_a|$, and $n_b = |V_b|$.

In a worst case distribution, a first place vote for candidates c_a or c_b cannot be altered, or else robustness is not guaranteed. Therefore, for any vote contained within $V_a \cup V_b$ has a minimum of $\frac{(m-1)!}{m!}$ safe permutations. It logically follows that all remaining votes $\notin V_a \cup V_b$ may be divided into two groups: the votes in which c_a is preferred to c_b, $V_{a \succ b}$, and the votes in which c_b is preferred to c_a, $V_{b \succ a}$.

For votes in $V_{a \succ b}$, any permutation that promotes c_a to first place is safe, of which there are $\frac{(m-1)!}{m!}$. Additionally, of the $\frac{(m-1)!}{m!}$ votes for which the existing first place candidate remains unchanged, exactly half maintain c_a's

relative position above c_b, and thus are also safe. Therefore, for votes contained in the set $V_{a \succ b}$, there are at least $\frac{3(m-1)!}{2m!}$ safe permutations.

For votes in $V_{b \succ a}$, it is again true that any permutation that promotes c_a to first place is safe, of which there are $\frac{(m-1)!}{m!}$. In this case, since c_b is preferred to c_a, a switch in their relative order only increases the pairwise score of c_a over c_b; therefore, all permutations which maintain the existing first place candidate are also safe, of which there are also $\frac{(m-1)!}{m!}$. It follows that for votes contained in the set $V_{b \succ a}$, there are at least $\frac{2(m-1)!}{m!}$ safe permutations.

With the inclusion of probabilities, the result is a lower bound of $\left(\frac{|V_a| + |V_b|}{n}\right)\frac{1}{m} + \left(\frac{|V_{a \succ b}|}{n}\right)\frac{3}{2m} + \left(\frac{|V_{b \succ a}|}{n}\right)\frac{2}{m}$. It is possible to improve this bound with further analysis by fixing the sizes of the vote sets without making any assumptions about the specific distribution. In order to minimize the bound, we maximize the multiplicative factor of the smallest term: $\frac{1}{m}$.

First we consider the size of V_a. If more than half the first place votes are for c_a, or $|V_a| > \frac{n}{2}$, it is clear that no other candidate may possibly beat c_a in either the plurality or runoff elections given only a single fault. It is straightforward to see that a distribution in which the outcome cannot be changed is obviously not a worst case distribution; therefore, we fix the size of set V_a at $\frac{n}{2}$.

Since it is a given that c_b loses to c_a in the pairwise runoff, it is clear that in a worst case, c_b must be challengeable by a third candidate who can win the runoff. In order for this to be possible, c_b may have no more than half of the remaining $\frac{n}{2}$ votes, so we fix the size of V_b at $\frac{n}{4}$.

With the smallest term, $\frac{1}{m}$, maximized, we consider the next smallest term: $\frac{3}{2m}$. To maximize this term, we can reasonably assign all remaining votes to it, fixing the remaining set sizes as $|V_{a \succ b}| = \frac{n}{4}$ and $|V_{b \succ a}| = 0$. Without making assumptions about the specific distribution other than it is a worst case, we have achieved a lower bound of $\frac{\frac{3n}{4}}{n} \cdot \frac{1}{m} + \frac{\frac{n}{4}}{n} \cdot \frac{3}{2m} = \frac{9}{8m}$.

The $\frac{1}{m}$ term used above is dependent upon the assumption that c_a is at risk of losing the plurality election; however, per the above fixed maximum

value, c_a is not challengeable. This allows us to reconsider the set sizes so that either c_a and c_b may be defeated in the plurality. To create an opportunity for a third candidate, we fix the size of $V_a \cup V_b$ as $\frac{2n}{3}$, and $V_{a \succ b}$ as $\frac{n}{3}$.

Given this distribution, there is the possibility for c_a to lose the plurality election, or for c_b to be replaced in the runoff by a candidate with a higher pairwise score than c_a, which are properties that follow logically for a worst case distribution. With these adjusted terms, we achieve a lower bound of $\frac{\frac{2n}{3}}{n} \cdot \frac{1}{m} + \frac{\frac{n}{3}}{n} \cdot \frac{3}{2m} = \frac{7}{6m}$. □

Chapter 4

Experimental Bounds on Robustness

4.1 Overview

In an attempt to provide more meaningful results for the robustness of these election rules, we conducted experiments that provide hard data for an average-case scenario. We wrote a program in Python that generates elections and randomly introduces noise to them. Each generated preference profile is run through each of six election rules: Borda, plurality, veto, Copeland, Maximin, Bucklin, and plurality with runoff. The winning candidate is recorded both before and after the noise fault is introduced, and the robustness is recorded. The resulting data is an average robustness for each election rule. The source code for the experiments can be found in Appendix B.

More specifically, the program generates elections in succession for all values of n (the number of voters) and m (the number of candidates) such that $10 \leq n \leq 250$ and $3 \leq m \leq 20$. The generated preference profiles are one of two types, per user input: random or "adversarial." The random profile is a set of cardinality n containing random permutations of $\langle 1, 2, \ldots, m \rangle$. The adversarial profile is a set of cardinality n that contains successive left circular shifts of $\langle 1, 2, \ldots, m \rangle$, i.e. $\{\langle 1, 2, \ldots, m \rangle, \langle 2, 3, \ldots, m, 1 \rangle, \langle 3, 4, \ldots, m, 1, 2 \rangle, \ldots\}$.

(Note that adversarial profiles for which $m \mid n$ have the property that the pairwise score of any two candidates is equal.)

The purpose of the adversarial profile is to approximate a worst case. While no one profile can serve as the worst case for all six of these rules, it seems logical that an election in which all the candidates' scores are equal must be close to a worst case, as it is easy to shift the outcome of the election with a single shift in the scores. The adversarial profile also serves as a "control" against the random elections, since their makeup is consistent across all values of n and m, and not dependent on a random number generator.

After each preference profile is generated, the winners of each of the six election rules are recorded for future comparison, and 100 faults (noise) are introduced independent of one another; that is, the noise is not successive or compounded, but rather the fault is removed from the profile before the next is introduced. The noise is either an arbitrary reordering or an elementary transposition, per user input, but is always added to a randomly selected vote (and for the transposition, performed at a randomly chosen point within the vote). Each of the election rules is conducted again on the altered profile, and if the outcome of the election is unchanged when compared to the outcome of the initial unaltered election, a score of 1 is added to the running score for that election rule. The fault is then removed (and thus the original profile restored) before the next fault is introduced. It should be noted that for the cases in which an election results in a tie, we declare the candidate with the lowest index to be the winner, in the interest of selecting an arbitrary candidate in a consistent fashion.

Following the completion of the 100 independent faults, the election is given a robustness score for each election rule by dividing the number of faults that maintained robustness by the total number of faults—the average robustness for that specific election, per election rule. This is the data point that is stored for later data analysis. The entire above process repeats once for each (n, m) pair, a total of 4,080 elections.

Once each of the elections has been conducted with 100 independent faults, the result is 100 distinct data points per election rule, each consisting of the rule, n, m, and the average robustness for a profile of those dimensions. For each rule, we find the mean robustness, the minimum and maximum robustness scores (over all 100 data points), the standard deviation from the mean, and the percentage of data points that are within plus or minus

one standard deviation from the mean. This analysis is the data found in the tables below, where the mean is μ, the minimum and maximum over all data points are $\geq$ and $\leq$ respectively, the standard deviation is σ, and the percentage within one standard deviation is $\pm 1\sigma$.

The algorithm described above is formally defined below in Figure 4.1.

$rules \leftarrow (Borda, plurality, veto, Copeland, Maximin, Bucklin, pluralitywithrunoff)$

$elections \leftarrow 0$
for all n such that $10 \leq n \leq 250$ **do**
  for all m such that $3 \leq n \leq 20$ **do**
    $winners \leftarrow []$
    $robustness \leftarrow []$
    for each $rule$ in $rules$ **do**
      $winners[rule] \leftarrow rule(\succ^V)$
    end for
    $scores \leftarrow []$
    $faults \leftarrow 0$
    while $faults < 100$ **do**
      $\succ^{V'} \leftarrow$ ADD-NOISE$(\succ^V)$
      for each $rule$ in $rules$ **do**
        $winner \leftarrow rule(\succ^{V'})$
        if $winner = winners[rule]$ **then**
          $scores[rule] \leftarrow scores[rule] + 1$
        end if
      end for
      $faults \leftarrow faults + 1$
    end while
    for each $rule$ in $rules$ **do**
      $robustness[rule] \leftarrow robustness[rule] \cup (scores[rule]/100)$
    end for
  end for
end for

Figure 4.1: The algorithm used to determine our experimental results.

4.2 Results for Arbitrary Reordering

To complement our own research of the arbitrary reordering noise type, we ran experiments for both random preference profiles and adversarial preference profiles. The resulting data is provided in Table 4.1 for random profiles, and Table 4.2 for adversarial profiles.

Rule	μ	$\geq$	$\leq$	σ	$\pm 1\sigma$
Borda	0.877	0.250	1.000	0.175	0.80
Plurality	0.913	0.400	1.000	0.115	0.81
Veto	0.921	0.360	1.000	0.097	0.84
Copeland	0.845	0.130	1.000	0.189	0.82
Maximin	0.882	0.120	1.000	0.177	0.80
Bucklin	0.903	0.080	1.000	0.133	0.82
Plurality w. Runoff	0.879	0.280	1.000	0.144	0.81

Table 4.1: Arbitrary reordering robustness results for random elections with 100 faults each. The mean is denoted by μ, the minimum and maximum over all data points by $\geq$ and $\leq$, standard deviation by σ, and percent within one standard deviation by $\pm 1\sigma$.

Rule	μ	$\geq$	$\leq$	σ	$\pm 1\sigma$
Borda	0.502	0.020	0.850	0.177	0.66
Plurality	0.214	0.010	0.660	0.129	0.77
Veto	0.502	0.040	0.960	0.222	0.60
Copeland	0.672	0.000	1.000	0.283	0.48
Maximin	0.523	0.010	1.000	0.192	0.85
Bucklin	0.271	0.010	1.000	0.159	0.72
Plurality w. Runoff	0.303	0.070	0.680	0.128	0.67

Table 4.2: Arbitrary reordering robustness results for adversarial elections with 100 faults each. The mean is denoted by μ, the minimum and maximum over all data points by $\geq$ and $\leq$, standard deviation by σ, and percent within one standard deviation by $\pm 1\sigma$.

4.3 Results for Elementary Transposition

For the purposes of comparison with previous work, for which no experimental data is available, we also conducted experiments using a single elementary transposition as the type of noise, rather than the arbitrary reordering. The parameters are all identical to the experiments above, but with a single randomly-placed transposition in the preference profile. The results can be found in Table 4.3 for random profiles and Table 4.4 for adversarial profiles.

Rule	μ	$\geq$	$\leq$	σ	$\pm 1\sigma$
Borda	0.984	0.290	1.000	0.070	0.94
Plurality	0.985	0.610	1.000	0.036	0.93
Veto	0.986	0.660	1.000	0.033	0.92
Copeland	0.987	0.390	1.000	0.037	0.93
Maximin	0.994	0.390	1.000	0.024	0.95
Bucklin	0.985	0.610	1.000	0.036	0.93
Plurality w. Runoff	0.984	0.570	1.000	0.037	0.92

Table 4.3: Elementary transposition robustness results for random elections with 100 faults each. The mean is denoted by μ, the minimum and maximum over all data points by $\geq$ and $\leq$, standard deviation by σ, and percent within one standard deviation by $\pm 1\sigma$.

Rule	μ	$\geq$	$\leq$	σ	$\pm 1\sigma$
Borda	0.864	0.010	1.000	0.269	0.88
Plurality	0.883	0.520	1.000	0.082	0.78
Veto	0.924	0.490	1.000	0.080	0.87
Copeland	1.000	0.610	1.000	0.006	1.00
Maximin	0.217	0.000	1.000	0.312	0.87
Bucklin	0.888	0.520	1.000	0.087	0.73
Plurality w. Runoff	0.896	0.380	1.000	0.078	0.85

Table 4.4: Elementary transposition robustness results for adversarial elections with 100 faults each. The mean is denoted by μ, the minimum and maximum over all data points by $\geq$ and $\leq$, standard deviation by σ, and percent within one standard deviation by $\pm 1\sigma$.

Chapter 5

Conclusion

A summary of our research results on the bounds of robustness for arbitrary reordering, compared with the Procaccia et al. [27] results for elementary transpositions, can be found in Table 3.1. It is unsurprising to find that the bounds on robustness for some rules under arbitrary reordering are significantly worse than the Procaccia et al. bounds on elementary transpositions; however, it is certainly noteworthy that the upper bounds on Copeland and Maximin are actually better for arbitrary reordering than for elementary transposition. As discussed above in the section 3.2 definitions, an arbitrary reordering is equivalent to a series of elementary transpositions confined to a single vote. It is straightforward to see that, in general, the 1-robustness of an arbitrary reordering is linked to the k-robustness of the elementary transposition.

The maximum number of elementary transpositions required to equate a single arbitrary reordering is the case in which a ranked preference list is completely inverted. For such a scenario, $\frac{m^2-m}{2}$ transpositions are required (see proof of Proposition 1 of Section 3.2). Since the 1-robustness of an arbitrary reordering is equivalent to the k-robustness of an elementary transposition, where k is bounded by $\frac{m^2-m}{2}$, the robustness of the arbitrary reordering is lower by a factor of nearly m^2. Furthermore, the difference increases quadratically as m, the number of candidates, increases.

It was interesting to obtain experimental data on the robustness of these elec-

tion rules, as we were unable to find previous work that does so. The results are not necessarily surprising, though there are certainly many noteworthy points worth discussion. Overall, it is reasonable to see that the robustness scores are much higher than the lower bounds provided in both our work and Procaccia et al. [27]. The experimental results are more of an average case; in any random preference profile, it is obviously extremely unlikely to have generated the worst case. Additionally, as would be expected, the robustness scores for the elementary transposition are significantly higher than for an arbitrary reordering.

The impetus for using a completely random preference profile was not to simulate a "realistic" election; indeed, in a real-world political election, for example, votes follow certain predictable patterns and trends, and are not so chaotic as to be completely random. The goal was simply to provide a glimpse at the robustness for an average, simple case. In other words, if you arbitrarily choose a single preference profile from the set of all profiles, how robust is it? While it does not provide a realistic election stimulation, it does provide a reasonable metric of how robust the voting rules are in comparison with one another.

Aside from the robustness scores of the random profiles are generally higher for elementary transposition than for arbitrary reordering, as was expected, there are several differences. In comparison among rules, the scores for elementary transposition are much more densely clustered, with a spread of just over 2%, while the spread for arbitrary reordering is well over 9%. The maximum value in every case pertaining to random profiles is 100%, and the minimums have a comparable spread between noise types, with one exception: the minimum Borda robustness for elementary transpositions is markedly lower than the other elementary transposition minimums. The reason for this is unclear.

Perhaps the most obvious and relevant difference between the types of noise is the data related to standard deviation: the standard deviations for elementary transpositions are extremely low in comparison to those for arbitrary reordering, often below 3%. The percentage within one standard deviation is also much higher, which follows logically from having a lower standard deviation. Overall, this indicates there is much less variance in the robustness of elementary transpositions. Due to the nature of the aforementioned relationship between the two types of noise, it is reasonable that the robustness of an arbitrary reordering varies to a much greater degree.

The purpose of the "adversarial" preference profiles was to investigate the robustness in an election where the candidates' scores are tied, or nearly so. For some voting rules, this may be the actual worst case, but regardless, it straightforward to see that this model of election will be more consistently susceptible to noise than a random election would be. As expected, the robustness scores for the adversarial profiles are markedly lower than those of the random profiles.

During the course of our research on the arbitrary reordering, we conceived another third type of noise that we had not previously considered, but that may be interesting to study: a noncontiguous swap. While an elementary transposition swaps the rank of two adjacent candidates in a preference list, a noncontiguous swap does not require the candidates be adjacent, only within the same vote. Such swaps became relevant in counting the safe permutations of an arbitrary reordering, and serves as a sort of "middle ground" between elementary transposition and arbitrary reordering. Further investigation into this type of noise may yield worthwhile results.

Appendix A

Suppression

We originally intended to investigate two different forms of noise: both the arbitrary reordering use throughout the work above, and also *suppression*. The concept of suppression is simply when an entire vote is "lost," or removed from the preference profile. Ultimately we directed our research away from suppression and focused solely on arbitrary reordering. The primary reason for this was that it quickly became apparent that it was difficult to obtain meaningful results for suppression. In a worst case scenario, the loss of an entire vote will always affect the election outcome, resulting in a robustness score of 0. In an average case, we do not expect the suppression of a vote to be very significant, since no candidate's score will increase as a result. We initially considered revisiting the notion of suppression following our work on reordering, but given the time constraints and scope of this particular work, it became infeasible. Included here is the preliminary research we conducted regarding suppression.

A.1 Definitions

A type of noise we call *suppression* occurs when a voter's entire ordinal preference list is removed from the preference profile and not counted at all. While this cannot strictly be achieved via elementary transpositions, it is an equally probable form of data corruption in the voting environment. A

suppression is defined as follows:

Definition 1. Given a preference profile $\succ^V$ and a voter $v^i \in V$, the preference profile $\succ^{V^i}$ is obtained from $\succ^V$ by a *suppression* if the following hold:

1. $V^i = V - \{v^i\}$
2. $\succ^{V^i} = \langle \succ^1, \ldots, \succ^{i-1}, \succ^{i+1}, \ldots, \succ^n \rangle$

The reader may be concerned with the sequential indexing problem that results from suppressing a voter and the voter's ordinal preferences. For the sake of simplicity, we assume that the voter indices are renumbered following the suppression, i.e. suppressing voter i from $V = \{v^1, v^2, \ldots, v^n\}$ yields $V^i = \{v^1, \ldots, v^{i-1}, v^{i+1}, \ldots, v^n\}$ which is renumbered to $V^i = \{v^1, \ldots, v^{n-1}\}$. The same is true for $\succ^V$, i.e. $\succ^{V^1} = \langle \succ^i, \ldots, \succ^{n-1} \rangle$.

Definition 2. The *k-robustness* of a preference profile $\succ^V$ under suppression of k voters $v^i \in V$ for $i \in \{1, \ldots, n\}$, written $\gamma_k(\succ^V) = \succ^{V^k}$, is:

$$\rho_{\gamma_k}(F, \succ^V) = \Pr_{\succ^{V^k} \sim \gamma_k(\succ^V)} \left[F(\succ^V) = F(\succ^{V^k}) \right].$$

Definition 3. The *k-robustness* of a voting rule F under suppression of k voters $v^i \in V$ for $i = 1, \ldots, k$ with n voters and m candidates is:

$$\rho^{n,m}_{\gamma_k}(F) = \min_{\succ^{V^k} \in \mathcal{L}(C)^n} \rho_{\gamma_k}(F, \succ^{V^k}).$$

A.2 Theorems

Since the idea of suppression is not a straightforward adaptation of the elementary transposition, we explicitly prove that a suppression can alter the outcome of an election:

Theorem 1. *Let $F : \mathcal{L}^V \to C$ be a voting rule such that $Ran(F) > 1$. Then there exists a preference profile $\succ^V$, a voter $v^i \in V$, and a profile $\succ^{V^i}$, which is obtained from $\succ^V$ by a suppression, such that $F(\succ^V) \neq F(\succ^{V^i})$.*

Proof. Since $Ran(F) > 1$, there must exist two preference profiles $\succ_1^V$ and $\succ_2^V$ such that $F(\succ_1^V) \neq F(\succ_2^V)$. Furthermore, all preference profiles are series of permutations on C and any preference profile can be obtained from another by iterative elementary transpositions. It follows that $\succ_2^V$ can be obtained from $\succ_1^V$ by a series of elementary transpositions with t intermediate states such that $\succ_{i_1}^V \rightsquigarrow \ldots \rightsquigarrow \succ_{i_t}^V$ where $\succ_{i_1}^V = \succ_1^V$ and $\succ_{i_t}^V = \succ_2^V$.

Since $F(\succ_1^V) \neq F(\succ_2^V)$, it also follows that there exists a pivotal intermediate state $\succ_{i_h}^V$ such that $F(\succ_{i_h}^V) = F(\succ_1^V)$ and $F(\succ_{i_{h+1}}^V) = F(\succ_2^V)$, and transitively $F(\succ_{i_h}^V) \neq F(\succ_{i_{h+1}}^V)$. Given the suppression of the voter $v^i \in V$ whose ordinal preferences were transposed in $\succ_{i_h}^V \rightsquigarrow \succ_{i_{h+1}}^V$, it holds true that $\succ_{i_h}^{V^i} = \succ_{i_{h+1}}^{V^i}$. However, since $F(\succ_{i_h}^V) \neq F(\succ_{i_{h+1}}^V)$, it must be the case that $F(\succ_{i_h}^{V^i}) \neq F(\succ_{i_h}^V)$ or $F(\succ_{i_{h+1}}^{V^i}) \neq F(\succ_{i_{h+1}}^V)$. □

Proposition 2. *Given k suppressions, $\rho_{\gamma_k}^{n,m}(F) \geq (\rho_{\gamma_1}^{n,m}(F))^k$.*

Proof. Consider the preference profile $\succ_1^V$ and the preference profile $\succ_2^{V^k}$ obtained from it by k independent and random suppressions. We claim the probability that $F(\succ_1^V) = F(\succ_2^{V^k})$ is at least $(\rho_{\gamma_1}^{n,m})^k$. Let $\succ_{i_1}^V, \ldots, \succ_{i_{k+1}}^{V^k}$ be the intermediate preference profiles obtained by the suppressions, where $\succ_{i_1}^V = \succ_1^V$, $\succ_{i_{k+1}}^{V^k} = \succ_2^V$, and each $\succ_{i_{j+1}}^{V^{n-j}}$ is obtained from $\succ_{i_j}^{V^{n-j-1}}$ by a random and independent suppression for $j = 1, \ldots, k$. By the definition of 1-robustness, for every preference profile $\succ^{V^i}$, the probability that one randomly chosen suppression does not change the outcome of the election under F is at least $\rho_{\gamma_1}^{n,m}(F)$. Therefore, for $j = 1, \ldots, k$,

$$\Pr\left[F(\succ_{i_j}^{V^{n-j-1}}) = F(\succ_{i_{j+1}}^{V^{n-j}}) \mid \succ_{i_j}^{V^{n-j-1}}\right] \geq \rho_{\gamma_1}^{n,m}(F).$$

Therefore:

$$\begin{aligned}\Pr\left[F(\succ_1^V) = F(\succ_2^V)\right] &= \prod_{j=1}^{k} \Pr\left[F(\succ_{i_j}^{V^{n-j-1}}) = F(\succ_{i_{j+1}}^{V^{n-j}}) \mid \succ_{i_j}^{V^{n-j-1}}\right] \\ &\geq (\rho_{\gamma_1}^{n,m})^k.\end{aligned}$$

□

Appendix B

Source Code

Below is the source code for the program used to conduct the experiments. It is written to run in version 2.5 of the Python interpreter.


```
"""Conducts a number of different elections based on well-defined and
studied election rules.

The preference profile should always be represented as an n x m matrix,
where n is the number of voters and m is the number of candidates. It
is essentially a list of all voters, where each voter is represented by
a list of candidates in preference order.
"""
```

```
import random
import sys
import copy
import csv
import optparse
from decimal import Decimal


class vote(object):
    """Represents a single voter's ordinal preferences.

    All indices are 0-based, e.g. the highest-ranked candidate is in
    position 0.
    """

    # The ordinal preference list
    _pref = []

    # Reverse mapping of candidate to ranking
    # e.g. _index[2] = 0 means candidate 2 is ranked 1st
    _index = []

    def __init__(self, prefOrSize):
        """Constructor takes either an ordinal preference list, or an int
        representing the length of a list to be randomly generated.
        """
        if type(prefOrSize) == int:
            # initialize to ascending list
            self._pref = range(prefOrSize)
            # and scramble it
            random.shuffle(self._pref)
        elif type(prefOrSize) == list:
            # Empty slice creates a copy of the list.
            # This is important since lists are passed by reference.
            self._pref = prefOrSize[:]
        else:
            # Only takes list or int. Could take some other iterables
            # e.g. tuple, but order matters and thus not all iterables
            # are acceptable.
            raise TypeError('Must be preference list or int size of random ' \
                            'vote.')

        # Initialize index mapping to 0's
        self._index = [0] * self.size()
```

```
    for x in xrange(self.size()):
        # Index maps candidate number to ranking
        self._index[self._pref[x]] = x

def scramble(self):
    """Randomly reorders the preference list of the vote."""
    random.shuffle(self._pref)
    # Must update index after scrambling, like at creation time.
    for x in xrange(self.size()):
        self._index[self._pref[x]] = x

def transposition(self):
    """Performs a random elementary transposition."""
    # Select a random point for the transposition
    p = random.randint(0, len(self._pref) - 2)
    # Swap p and p+1
    tmp = self._pref[p]
    self._pref[p] = self._pref[p + 1]
    self._pref[p + 1] = tmp

    # Must update index after transposition, like at creation time.
    for x in xrange(self.size()):
        self._index[self._pref[x]] = x

def size(self):
    """Returns the length of the preference list."""
    return len(self._pref)

def rank(self, candidate):
    """Returns the rank of the given candiate in this vote's preference
    list.
    """
    return self._index[candidate]

def pos(self, position):
    """Returns the candidate with the given rank.

    This is equivlent to calling getitem with square brackets.
    """
    return self._pref[position]

def __getitem__(self, key):
    """Overload accesses candidate in the given position."""
    if type(key) != int:
        # Support for slices could be added easily, but there is no
```

```
            # need at this point in time.
            raise TypeError('Only int keys are allowed, no slices.')
        return self.pos(key)


def winner(score, top=1):
    """Determines the winner of an election given a list of the
    candidates' scores.

    The score parameter is to be a list of numerical scores, ordered by
    candidate index. The scale of scores is irrelevant, as they are
    compared only relative to each other.

    The top parameter will return all the candidates ranked in the top
    n places, including all ties.

    The function returns a tuple of the indices of the winning
    candidates (even if there is only one).
    """
    # Creates pairs of (candidate,score),
    # then sorts by score highest to lowest
    # Important note: the secondary sort order is by index, since they
    # are first sorted as such by the virture of the enumeration
    # This comes into play for 'arbitrary' selection of ties using the
    # minimum index. (Stable sort guaranteed by Python 2.3 and later.)
    index = list(enumerate(score))
    index.sort(key=lambda x: x[1])

    wins = []  # the list of winners
    last = -1  # previous candidates score, used to find ties
    go = True

    # Loops until top n candidates are found, including ties
    while go:
        next = index.pop() # next highest scoring candidate
        if len(wins) < top or next[1] == last:
            # Haven't met "top" req,
            # Or this candidate tied with the last one
            wins.append(next[0])
            last = next[1]
        else:
            # Otherwise, wins is complete and we stop the loop
            go = False
        if not index:
            # There are no more scores, stop the loop
```

```
            go = False

    return tuple(wins)

def scoringRule(alpha, votes, top=1):
    """Scores candidates given a preference profile and an alpha
    vector, and returns the indices of the winners.

    This is intended as a helper function for "scoring rule" functions,
    but could be used separately if needed.

    The alpha vector, which must be the same size as the candidate list,
    provides a numberical score for each rank.

    The votes parameter is an n x m matrix representing the preference
    profile.

    The top parameter is a pass-through to the winner() function.
    """
    size = len(alpha)
    score = [0] * size

    # Detect malformed preferences or alpha vector
    for v in votes:
        if v.size() != size:
            raise IndexError('All votes must be the same size as the alpha ' \
                             'vector.')

    # Adds the alpha vector value to the score list based on rank in
    # the preference profile
    for i,s in enumerate(alpha):
        for v in votes:
            score[v[i]] += s

    return winner(score, top=top)

def pairwise(c1, c2, votes):
    """Conducts a pairwise election between given candidates c1 and c2.

    votes must be the n x m preference profile matrix.
    """
    score = [0,0]

    # Simply adds 1 for each time a candidate is ranked above the other
    for v in votes:
```

```
        if v.rank(c1) < v.rank(c2):
            score[0] += 1
        else:
            score[1] += 1

    return tuple(score)

def borda(votes):
    """Conducts a Borda election using the scoringRule() function.

    votes must be the n x m preference profile matrix.
    """
    # Alpha vector is {m, m-1, m-2, ..., 0}
    alpha = range(votes[0].size() - 1, -1, -1)
    return scoringRule(alpha, votes)

def plurality(votes):
    """Conducts a plurality election using the scoringRule() function.

    votes must be the n x m preference profile matrix.
    """

    # Alpha vector is {1, 0, 0,..., 0}
    alpha = [0] * votes[0].size()
    alpha[0] = 1
    return scoringRule(alpha, votes)

def plurality2(votes):
    """Conducts a "top 2" plurality election using the scoringRule()
    function.

    votes must be the n x m preference profile matrix.
    """
    # Same vector is plurality, but has top=2 param
    alpha = [0] * votes[0].size()
    alpha[0] = 1
    return scoringRule(alpha, votes, 2)

def veto(votes):
    """Conducts a veto election using the scoringRule() function.

    votes must be the n x m preference profile matrix.
    """
    # Alpha vector is {1, 1, ..., 1, 0}
    alpha = [1] * votes[0].size()
```

```
    alpha[-1] = 0
    return scoringRule(alpha, votes)

def copeland(votes):
    """Conducts a Copeland election.

    votes must be the n x m preference profile matrix.
    """
    size = votes[0].size()
    score = [0] * size

    # Conducts pairwise elections for every candidate pairing
    for c1 in xrange(size):
        for c2 in (c for c in xrange(size) if c != c1):
            pw = pairwise(c1, c2, votes)
            if pw[0] > pw[1]:
                score[c1] += 1
            # No 'else' here because c2 < c1 case is handled
            # in the opposite iteration

    return winner(score)

def maximin(votes):
    """Conducts a Maximin (Minimax) election.

    votes must be the n x m preference profile matrix.
    """
    size = votes[0].size()
    score = [size**2] * size

    for c1 in xrange(size):
        for c2 in (c for c in xrange(size) if c != c1):
            pw = pairwise(c1, c2, votes)[0]
            if pw < score[c1]:
                score[c1] = pw

    return winner(score)

def bucklin(votes):
    """Conducts a Bucklin election.

    votes must be the n x m preference profile matrix.
    """
    size = votes[0].size()
    for l in xrange(size):
```

```
        # l is the deepest level we look at
        # Proceed deeper until a majority is reached
        # Start with blank scores for each new level
        score = [0] * size
        for j in xrange(size):
            for v in votes:
                if v.rank(j) <= l:
                    # Candidate j is ranked above l threshold
                    score[j] += 1
        if max(score) > (size / 2):
            # Majority has been achieved, terminate
            break

    return winner(score)


def pwr(votes):
    """Conducts a Plurality with Runoff election.

    votes must be the n x m pereference profile matrix.
    """
    # Round 1: top 2 plurality
    c1, c2 = plurality2(votes)[:2]
    # Round 2: pairwise runoff
    s1, s2 = pairwise(c1, c2, votes)
    if s1 > s2:
        return (c1,)
    elif s2 > s1:
        return (c2,)
    else:
        return (c1, c2)


def calc_stats(data):
    """Given a list of robustness scores, calculates the mean, standard
    deviation from the mean, and precentage of points within +/- 1
    standard deviation from the mean.

    Returns tuple of mean, standard deviation, within 1 std deviation.
    """
    n = Decimal(len(data))

    # First find the mean
    total = sum(data)
    mean = total / n
```

```
    # Calculate the average variance (square of the std dev)
    variance = 0
    for p in data:
        # Variance is the square of distance from the mean
        variance += (p - mean) ** 2
    variance /= n

    # Calculate the std dev
    stdev = variance.sqrt()

    # Use std dev to find % within 1 std dev
    w1stdev = 0
    for p in data:
        # Difference from the mean is within the std dev
        if abs(p - mean) <= stdev:
            w1stdev += 1
    w1stdev /= Decimal(n)

    return mean, stdev, w1stdev

def random_profile(n, m):
    """Generates a random preference profile using the given
    constraints.

    n is the number of voters, and m is the number of candidates.
    """
    votes = []
    for j in xrange(n):
        # vote ctor makes random profile by default
        v = vote(m)
        votes.append(v)

    return votes


def adversarial_profile(n, m):
    """Generates an adversarial preference profile using the given
    constraints.

    The adversarial profile is successive left circular shifts of the
    profile <1,2,...,m>, e.g. {<1,2,...,m>, <2,3,...,m,1>, ...}.

    n is the number of voters, and m is the number of candidates.
    """
```

```
    votes = []
    # The initial <1,2,...,m> profile
    base = range(m)

    for i in xrange(n):
        # Have to use copies because the list is a reference
        nvote = vote(copy.copy(base))
        votes.append(nvote)
        # Move first candidate to end
        t = base.pop(0)
        base.append(t)

    return votes

def election(rule, votes, one=True):
    """Conducts an election given the election rule and preference profile.

    rule is a reference to one of the above election rules, and votes
    is a standard preference profile.

    The optional one parameter specifies whether ties must broken.
    """
    wins = rule(votes)
    if one or len(wins) == 1:
        # Restrict to 1 candidate by arbitrary selection of
        # min index OR
        # There was only 1 winner anyway
        wins = wins[0]

    return wins

def nm_gen():
    """Generates all valid pairs of (n,m), where 10 <= n <= 250 and 3 <= m <= 20.
    """
    n = 10
    m = 3

    while m <= 20:
        yield n, m
        n += 1
        if n > 250:
            n = 10
            m += 1
```

```
if __name__ == '__main__':
    usage = """usage: %prog profile-type noise-type [-e elections] [-f faults] [-o filename]
[--ties]

profile-type may be either "random" for randomly generated elections, or
"adversarial" for elections of the adversarial type. Abbreviations accepted.

noise-type may be either "ar" for arbitrary reordering, or "et" for elementary
transposition.

Try -h or --help for more information."""
    parser = optparse.OptionParser(usage=usage)
    parser.add_option('-f', dest='faults', type='int', default=100,
                      help='the number of faults to introduce per election, default 100')
    parser.add_option('-o', dest='filename', default='vote-data.csv',
                      help='the name of the data output file, default ' \
                      '"vote-data.csv".')
    parser.add_option('--ties', action='store_false', dest='one', default=True,
                      help='returns all ties as winners, default is to select' \
                           ' a single winner')
    (options, args) = parser.parse_args()

    if len(args) < 2:
        parser.error('incorrect number of arguments')

    if 'random'.startswith(args[0].lower()):
        profile_create = random_profile
    elif 'adversarial'.startswith(args[0].lower()):
        profile_create = adversarial_profile
    else:
        parser.error('invalid profile type')

    if args[1].lower() == 'ar':
        noise_type = 0
    elif args[1].lower() == 'et':
        noise_type = 1
    else:
        parser.error('invalid noise type')

    rules = ('borda', 'plurality', 'veto', 'copeland', 'maximin', 'bucklin', 'pwr')

    # 0 func, 1 robust, 2 min, 3 max, 4 current winner, 5 list of scores
    data = {}
    for func in rules:
        # default values
```

```
    data[func] = [eval(func), 0, 1, 0, None, []]

fout = open(options.filename, 'wb')
csvout = csv.writer(fout, dialect='excel')

gen = nm_gen()  # generates all valid pairs of (n,m)
for n,m in gen:
    # Creates the list of n votes, fills with random vote of size m
    votes = profile_create(n, m)

    for rule in rules:
        # call election rule to get winner(s) and record it for
        # future comparison
        data[rule][4] = election(data[rule][0], votes, options.one)

    # for each required fault
    for j in xrange(options.faults):
        # pick a random vote (by index)
        ri = random.randrange(n)
        # keep a copy of the original
        orig = copy.deepcopy(votes[ri])
        # introduce noise
        if noise_type == 0:
            votes[ri].scramble()
        elif noise_type == 1:
            votes[ri].transposition()

        for rule in rules:
            # see if the outcome changed for any of the rules
            if election(data[rule][0], votes, options.one) == data[rule][4]:
                # outcome did not change, add 1 to robustness score
                data[rule][1] += 1
        # replace the scrambled vote with the original
        votes[ri] = orig

    for rule in rules:
        # calculate robustness % score
        robust = data[rule][1] / Decimal(options.faults)
        data[rule][5].append(robust)
        data[rule][1] = 0
        # update max and min
        if robust < data[rule][2]:
            data[rule][2] = robust
        elif robust > data[rule][3]:
            data[rule][3] = robust
```

```
        csvout.writerow((rule, n, m, robust))

fout.close()

head = '  Rule     Mean    Min    Max   Std Dev  <= 1 SD'
print head
print '-' * len(head)
for rule in rules:
    mean, stdev, w1stdev = calc_stats(data[rule][5])
    dp = (rule.capitalize(), mean, data[rule][2], data[rule][3], stdev, w1stdev)
    print '%-9s  %-4.3f  %-3.3f  %-3.3f   %-6.3f   %.2f' % dp
```

References

[1] K. J. Arrow. *Social Choice and Individual Values*. Yale University Press, 2nd edition, 1963.

[2] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[3] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

[4] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. How hard is it to control en election? Technical Report 89-90-07, Georgia Institute of Technology, 1990.

[5] D. Black. On the rationale of group decision-making. *The Journal of Political Economy*, 56(1):23–34, Feb. 1948.

[6] S. J. Brams and P. C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare*, chapter 4. North-Holland, 2002.

[7] J. W. Bucklin. The Grand Junction plan of city government and its results. *Annals of the American Academy of Political and Social Science*, 38(3):87–102, Nov. 1911.

[8] J. R. Chamberlin, J. L. Cohen, and C. H. Coombs. Social choice observed: Five presidential elections of the American Psychological Association. *The Journal of Politics*, 46(2):479–502, May 1984.

[9] V. Conitzer, J. Lang, and T. Sandholm. How many candidates are needed to make elections hard to manipulate? In *TARK '03: Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, pages 201–214, New York, NY, USA, 2003. ACM.

[10] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence*, pages 314–319, 2002.

[11] A. H. Copeland. A 'reasonable' social welfare function. mimeo, University of Michigan Seminar on Applications of Mathematics to the Social Sciences, 1951.

[12] U. Endriss and J. Lang, editors. *Proceedings of the 1st International Workshop on Computational Social Choice (COMSOC-2006)*. ILLC, University of Amsterdam, Dec. 2006.

[13] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. The complexity of bribery in elections. In Ulle Endriss and Jérôme Lang, editors, *Proceedings of the 1st International Workshop on Computational Social Choice (COMSOC-2006)*. ILLC, University of Amsterdam, Dec. 2006.

[14] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and Jörg Rothe. Copeland voting fully resists constructive control. Technical Report TR-2007-923, University of Rochester, Oct. 2007.

[15] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and Jörg Rothe. Llull and copeland voting broadly resist bribery and control. Technical Report TR-2007-913, University of Rochester, Feb. 2007.

[16] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.

[17] A. Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica*, 45(3):665–681, 1977.

[18] G. Hägele and F. Pukelsheim. Llull's writings on electoral systems. *Studia Lulliana*, 41:3–38, 2001.

[19] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Exact analysis of dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to np. *J. ACM*, 44(6):806–825, 1997.

[20] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Hybrid elections broaden complexity-theoretic resistance to control. *ArXiv Computer Science e-prints*, August 2006.

[21] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artif. Intell.*, 171(5-6):255–285, 2007.

[22] G. Kalai. Noise sensitivity and chaos in social choice theory. Discussion Paper Series dp399, Center for Rationality and Interactive Decision Theory, Hebrew University, Jerusalem, Aug. 2005.

[23] J. Kemeny. Mathematics without numbers. *Dædalus*, 88:571–591, 1959.

[24] J. Kemeny and L. Snell. *Mathematical Models in the Social Sciences*. Ginn & Co., Boston, 1960.

[25] G. H. Kramer. A dynamical model of political equilibrium. *Journal of Economic Theory*, 16(2):310–334, Dec. 1977.

[26] I. McLean and A. B. Urken, editors. *Classics of Social Choice*. The University of Michigan Press, Ann Arbor, 1995.

[27] A. D. Procaccia, J. S. Rosenschein, and G. A. Kaminka. On the robustness of preference aggregation in noisy environments. In Ulle Endriss and Jérôme Lang, editors, *Proceedings of the 1st International Workshop on Computational Social Choice (COMSOC-2006)*. ILLC, University of Amsterdam, Dec. 2006.

[28] T. C. Ratliff. A comparison of Dodgson's method and the Borda count. *Economic Theory*, 20(2):357–372, 2002.

[29] D. G. Saari. Susceptibility to manipulation. *Public Choice*, 64(1):21–41, 1990.

[30] P. B. Simpson. On defining areas of voter choice: Professor Tullock on stable voting. *The Quarterly Journal of Economics*, 83(3):478–490, Aug. 1969.

[31] H. P. Young and A. Levenglick. A consistent extension of Condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300, Sep. 1978.