7

Stackelberg Voting Games

Using computational complexity to protect elections from manipulation, bribery, control, and other types of strategic behavior is one of the major topics of Computational Social Choice. This raises the following fundamental question: Why should we prevent voters' strategic behavior? Of course we may answer this question by arguing that people should be sincere in voting due to ethical, sociological, political, or even divine reasons. However, after all, the most important objective of voting is to select a "good" alternative, especially in multi-agent systems. Therefore, we would prefer to give an answer that is similar to the following: we want to prevent voters' strategic behavior because it might lead to undesirable outcomes.

Showing evidence for this answer in the voting setting is not as simple as it may seem to be. One approach is to consider the game where all voters vote at the same time, and study the equilibria of this simultaneous-move voting game. Unfortunately, even in a complete-information setting where all voters' preferences are common knowledge, this leads to an extremely large number of equilibria, many of them bizarre. For example, as we have seen in an example in Section 3.2, in a plurality election with the lexicographic tie-breaking mechanism, it may be the case that

all voters' true preferences are Obama≻Clinton≻McCain. Nevertheless, the profile where all three voters vote for McCain≻Clinton≻Obama is a Nash equilibrium. This equilibrium is quite robust, because voting for Obama is a waste, given that nobody else is expected to vote for Obama and some votes went to Clinton. There has been some work exploring different solution concepts in simultaneous-move voting games—e.g., Farquharson (1969) and Moulin (1979)—but in some sense, the equilibrium selection issue in the above example is inherent in settings where voters vote simultaneously.

In many practical situations, the voters vote one after another, and the later voters know the votes cast by the earlier voters. For example, consider online systems that allow users to rate movies or other products. We consider the setting where the voters vote one after another in this chapter to overcome the equilibrium selection problem. We assume that voters' preferences over the alternatives are strict; we also make a complete-information assumption that the voters' preferences are common knowledge (among the voters themselves, though not necessarily to the election organizer).[1] This results in an extensive-form game of perfect information that can be solved by backward induction. In sharp contrast to the simultaneous-move setting, this results in a unique outcome (winning alternative). We refer to this game as a *Stackelberg voting game*.

Our main theoretical results will be shown in Section 7.2. As a corollary to our main theorem, which is quite technical but very general, we will show that for any voting rule r that satisfies the majority criterion (see Section 2.2 for the definition), no matter how many voters there are, there always exists a profile such that the backward-induction winner (i.e., the unique winner in all SPNE) of the Stackelberg

[1] While this is clearly a simplifying assumption, it approximates the truth in many settings, and with this assumption we do not need to specify prior distributions over preferences. Also, naturally, our negative results still apply to more general models, including models allowing for incomplete information, so long as the complete-information setting is a special case.

voting game that uses r is ranked within the bottom two positions in all voters' true preferences, with only two exceptions. This result is quite negative, because it says that if we allow voters to vote strategically, then sometimes the outcome is almost the *worst* outcome for all but two voters. Therefore, to some extent we are showing an ordinal *price-of-anarchy (PoA)* (Koutsoupias and Papadimitriou, 1999). The PoA is the ratio of the optimal social welfare over the worst social welfare in equilibrium outcomes. In fact, in the settings where social welfare is not well-defined, it is even not clear how the PoA should be defined. Fortunately, the paradoxes we will show are clearly very negative results.

Similar to the "worst case vs. typical case" debate about the results on hardness of manipulation, here again we can ask how often the paradoxes happen. To answer this question, we will pursue an empirical approach in Section 7.4. We will use the techniques developed in Section 7.3 to run simulations to compare the backward-induction winner to a benchmark outcome—namely, the alternative that would win if all voters voted truthfully. Our experimental results show that, surprisingly, more voters prefer the backward-induction outcome over the truthful outcome on average. Therefore, it seems that on average the backward-induction outcome is not too undesirable.

The idea of modeling a voting process in which voters vote one after another as an extensive-form game is not new. Sloth (1993) studied elections with two alternatives, as well as settings with more alternatives where a pairwise decision between two options is made at every stage. She relates the outcomes of this process to the multistage sophisticated outcomes of the game (McKelvey and Niemi, 1978; Moulin, 1979). In the extensive-form games studied by Dekel and Piccione (2000), multiple voters can vote simultaneously in each stage. They compare the equilibrium outcomes of these games to the outcomes of the *symmetric equilibria* of their simultaneous counterparts. Battaglini (2005) studies how these results are affected by the

possibility of abstention and a small cost of voting.

Our approach is significantly different from the previous approaches in several aspects. First, the prior work focuses mostly on the case of two alternatives or, in the case of multiple alternatives, on particular voting procedures; in contrast, we consider general (anonymous) voting rules with any number of alternatives, and correspondingly derive very general paradoxes. Second, we show some paradoxes to illustrate that the strategic behavior of the voters sometimes leads to very undesirable outcomes. Third, we also study how the backward-induction outcome can be efficiently computed, and we use these algorithmic insights in simulations to evaluate the quality of the Stackelberg voting game's outcome "on average."

Desmedt and Elkind (2010) simultaneously and independently studied a similar setting in which voters vote sequentially under the plurality rule, and showed several different types of paradoxes. In their model, voters are allowed to abstain, and voting comes at a small cost. They assume random tie-breaking and therefore need to consider expected utilities, while in our model studied in this chapter, voters' preferences are ordinal.

7.1 Stackelberg Voting Game

We now consider the strategic Stackelberg voting game. We use a complete-information assumption: all the voters' preferences are common knowledge. Given this assumption, for any voting rule r, the process where voters vote in sequence can be modeled as an extensive-form game of perfect information. In Section 3.2 we gave the formal definition of simultaneous-move voting games, and mentioned that extensive-form voting games can be defined similarly. Here I will be more specific. The game has n stages. In stage j ($j \leqslant n$), voter j chooses an action from $L(\mathcal{C})$. Each leaf of the tree is associated with an outcome, which is the winner for the profile consisting of the votes that were cast to reach this leaf.

Because the voters' preferences are linear orders (which implies that there are no ties), we can solve the game by backward induction, which results in a unique outcome. We note that this requires only ordinal preferences, that is, we do not need to define utilities. The backward-induction process works as follows. First, for any subprofile of votes by the voters 1 through $n-1$ (that is, any node that is the parent of leaves), there will be a nonempty subset of alternatives that n can make win by casting some vote. She will pick her most preferred one. Now, because we can predict what voter n will do, we take voter $(n-1)$'s perspective: for any subprofile of votes by the voters 1 through $n-2$, there will be a nonempty subset of alternatives that voter $n-1$ can make win by casting some vote (taking into account how voter n will act). She will pick her most preferred one; etc. We continue this process all the way to the root of the tree; the outcome there is called the *backward-induction outcome*.

As noted above, only the ordinal preferences of the voters matter; that is, a voter's preferences correspond to a member of $L(\mathcal{C})$. While votes and preferences both lie in the same set $L(\mathcal{C})$, we must be careful to distinguish between them, because in this context, a voter will sometimes cast a vote that is different from her true preferences. Nevertheless, we can use $P \in \mathcal{F}_n$ to denote a profile of preferences, as well as a profile of votes. For a given voting rule r, let $r(P)$ be the outcome if the *votes* are P; let $SG_r(P)$ be the backward-induction outcome if the *true preferences* are P.[2]

7.2 Paradoxes

In this section, we investigate whether the strategic behavior described above will lead to undesirable outcomes. It turns out that it can. Our main theorem is a

[2] Of course, because it is a function from profiles of linear orders to alternatives, SG_r can also be interpreted as a voting rule, though there is a significant risk of confusion in doing so. We note that even if r is anonymous, SG_r (as a voting rule) is not necessarily anonymous (the order of the voters matters).

general result that applies to many common voting rules. We will show that, for such a rule, there exists a profile that has two types of paradox associated with it in the backward-induction outcome: first, the winner loses all but one of its pairwise elections; second, the winner is ranked somewhere in the bottom two positions in almost every voter's true preferences. For the second type of paradox, we will show that the number of exceptions (voters who rank the winner higher) is closely related to a parameter called the *domination index*. The domination index of a voting rule r that satisfies non-imposition is the smallest number q such that any coalition of $\lfloor n/2 \rfloor + q$ voters can make any given alternative win (no matter how the remaining voters vote) under r. We note that the domination index is always well defined for any rule that satisfies non-imposition, and is at least 1.

Definition 7.2.1. *For any voting rule r that satisfies non-imposition, and any $n \in \mathbb{N}$, we let the* domination index *$DI_r(n)$ be the smallest number q such that for any alternative c, and for any subset of $\lfloor n/2 \rfloor + q$ voters, there exists a profile P for these voters, such that for any profile P' for the remaining voters, $r(P \cup P') = c$.*

The domination index DI_r is closely related to the *anonymous veto function* $\mathrm{VF}_r : \{1, \ldots, n\} \to \{0, \ldots, m\}$ (Definition 10.4 in Moulin (1991)), defined as follows. $\mathrm{VF}_r(i)$ is the largest number $j \leqslant m-1$ such that any coalition of i voters can veto any subset (that is, make sure that none of the alternatives in the set is the winner) of no more than j alternatives. We note that the domination index $\mathrm{DI}_r(n)$ for a voting rule r is the smallest number q such that $\mathrm{VF}_r(\lfloor n/2 \rfloor + q) = m - 1$ (that is, any coalition of size $\lfloor n/2 \rfloor + q$ can veto any set of $m-1$ alternatives).

The next proposition gives bounds on the domination index for some common voting rules.

Proposition 7.2.2. *For any positional scoring rule r, $DI_r \leqslant \lfloor n/2 \rfloor - \lfloor n/m \rfloor$. $DI_r(n) = 1$ for any voting rule r that satisfies the majority criterion (Section 2.2), including*

any rule that satisfies the Condorcet criterion (Section 2.2), plurality, plurality with runoff, Bucklin, and STV.

The next lemma provides a sufficient condition for an alternative not to be the backward-induction winner. It says that if there is a coalition of size $k \geqslant \lfloor n/2 \rfloor + \mathrm{DI}_r(n)$ who all prefer c to d, and another condition holds, then d cannot win.[3] For any alternative $c \in \mathcal{C}$ and any $V \in L(\mathcal{C})$, we let $\mathrm{Up}(c, V)$ denote the set of all alternatives that are ranked higher than c in V.

Lemma 7.2.3. *Let P be a profile. An alternative d is not the winner $SG_r(P)$ if there exists another alternative c and a sub-profile $P_k = (V_{i_1}, \ldots, V_{i_k})$ of P that satisfies the following conditions: 1. $k \geqslant \lfloor n/2 \rfloor + DI_r(n)$, 2. $c \succ d$ in each vote in P_k, 3. for any $1 \leqslant j_1 < j_2 \leqslant k$, $Up(c, V_{i_{j_1}}) \supseteq Up(c, V_{i_{j_2}})$.*

Proof. Let $D_k = \{i_1, \ldots, i_k\}$. Since $k \geqslant \lfloor n/2 \rfloor + \mathrm{DI}_r(n)$, this coalition of voters can guarantee that any given alternative be the winner under r, if they work together. Let $P_k^* = (V_{i_1}^*, \ldots, V_{i_k}^*)$ be a profile that can guarantee that c be the winner under r. That is, for any profile P' for the other voters ($\{1, \ldots, n\} \backslash D_k$), we have $r(P_k^*, P') = c$. For any $j \leqslant k$, we let $D'_{i_j} = \{1, \ldots, i_j\} \backslash D_k$—that is, the first i_j voters, except those in the coalition D_k. For any $j \leqslant k$, we let $P_j^* = (V_{i_1}^*, \ldots, V_{i_j}^*)$. That is, P_j^* consists of the first j votes in P_k^*. For any $i \leqslant n-1$ and any pair of profiles P_1 (consisting of i votes) and P_2 (consisting of $n-i$ votes), we let $SG_r(P_2 : P_1)$ denote the backward-induction winner of the subgame of the Stackelberg voting game in which voters 1 through i have already cast their votes P_1, and the true preferences of voters $i+1$ through n are as in P_2. We prove the following claim by induction.

[3] This may seem trivial because the coalition can guarantee that c wins if they work together. However, we have to keep in mind that the members of the coalition each pursue their own interest. For example, it may be the case that whenever the second-to-last voter in the coalition votes in a way that enables the last voter in the coalition to make c the winner, it also enables this last voter to make e the winner, which this last voter prefers—but the second-to-last voter actually prefers d to e, and therefore votes to make d win instead. We need the extra condition to rule out such examples.

Claim 7.2.1. *For any $j \leqslant k$ and any profile P'_{i_j} for the voters in D'_{i_j},*

$$SG_r((V_{i_j}, V_{i_j+1}, \ldots, V_n) : P'_{i_j}, P^*_{j-1}) \succeq_{V_{i_j}} c$$

Claim 7.2.1 states that for any $j \leqslant k$, if voters $i_1, \ldots, i_{j-1}$ have already voted as in P^*_j, and voter i_j will vote next, then the backward-induction outcome of the corresponding subgame must be (weakly) preferred to c by voter i_j.

Proof of Claim 7.2.1: The proof is by (reverse) induction on j. First, we consider the base case where $j = k$. If voter i_k casts $V^*_{i_k}$, then the winner is c, because the subprofile P^*_k will guarantee that c wins. Voter i_k will only vote differently if it results in at least as good an outcome for her as c. Therefore, the claim holds for $j = k$.

Now, suppose that for some j', the claim holds for $j' \leqslant j \leqslant k$. We will now show that it also holds for $j = j' - 1$. Let c' be the backward-induction outcome when voter $i_{j'-1}$ submits $V^*_{i_{j'-1}}$. By the induction hypothesis, we have that $c' \succeq_{V_{i_{j'}}} c$. That is, voter $i_{j'}$ (weakly) prefers c' to c. We recall that $\mathrm{Up}(c, V_{i_{j'-1}}) \supseteq \mathrm{Up}(c, V_{i_{j'}})$, which means that c' is also (weakly) preferred to c by voter $i_{j'-1}$. This means that voter $i_{j'-1}$ can guarantee that the outcome be at least as good as c for her. She will only vote differently from $V^*_{i_{j'-1}}$ if it results in at least as good an outcome for her as c' (which is at least as good as c already). Therefore, the claim also holds for $j' - 1$, and Claim 7.2.1 follows by induction. □

Letting $j = 1$ in Claim 7.2.1, we have that $SG_r(P) \succeq_{V_{i_1}} c$. Therefore, $d \neq SG_r(P)$ (because $c \succ_{V_{i_1}} d$). This completes the proof of Lemma 7.2.3. □

We are now ready to present our main theorem. We note that this theorem does not depend on the tie-breaking mechanism used in the rule.

Theorem 7.2.4. *For any voting rule r that satisfies non-imposition, and any $n \in \mathbb{N}$, there exists a profile P such that $SG_r(P)$ is ranked somewhere in the bottom two*

positions in $n - 2DI_r(n)$ of the votes, and, if $DI_r(n) < n/4$, then $SG_r(P)$ loses to all but one alternative in pairwise elections.

Proof. The proof is constructive. Let $P = (V_1, \ldots, V_n)$ be the profile (the voters' true preferences) defined as follows.

$$V_1 = \cdots = V_{\lfloor n/2 \rfloor - \mathrm{DI}_r(n)} = [c_3 \succ \ldots \succ c_m \succ c_1 \succ c_2]$$
$$V_{\lfloor n/2 \rfloor - \mathrm{DI}_r(n)+1} = \cdots = V_{\lfloor n/2 \rfloor + \mathrm{DI}_r(n)} = [c_1 \succ c_2 \succ c_3 \succ \cdots \succ c_m]$$
$$V_{\lfloor n/2 \rfloor + \mathrm{DI}_r(n)+1} = \cdots = V_n = [c_2 \succ c_3 \succ \cdots \succ c_m \succ c_1]$$

We now use Lemma 7.2.3 to prove that $SG_r(P) = c_1$. First, we let $k = \lfloor n/2 \rfloor + \mathrm{DI}_r(n)$, and let P_k be the first k votes. It follows from Lemma 7.2.3 (letting $c = c_1$ and $d = c_2$) that $c_2 \neq SG_r(P)$. Next, for any $c' \in \mathcal{C} \backslash \{c_1, c_2\}$, we let $k = \lceil n/2 \rceil + \mathrm{DI}_r(n)$ and let P_k be the last k votes, that is, $P_k = (V_{\lfloor n/2 \rfloor - \mathrm{DI}_r(n)+1}, \ldots, V_n)$. By Lemma 7.2.3 (letting $c = c_2$ and $d = c'$), we have that $c' \neq SG_r(P)$. It follows that $SG_r(P) = c$.

In P, c_1 is ranked somewhere in the bottom two positions in $n - 2\mathrm{DI}_r(n)$ votes (the first $\lfloor n/2 \rfloor - \mathrm{DI}_r(n)$ votes and the last $\lceil n/2 \rceil - \mathrm{DI}_r(n)$ votes). If $\mathrm{DI}_r(n) < n/4$, then $2\mathrm{DI}_r(n) < n/2$, which means that c_1 will lose to any other alternative (except c_2) in pairwise elections. □

Combining Proposition 7.2.2 and Theorem 7.2.4, we obtain the following corollary for common voting rules.

Corollary 7.2.5. *Let r be any rule that satisfies non-imposition and majority criterion, and let $n \geqslant 5$. There exists a profile P such that $SG_r(P)$ is ranked somewhere in the bottom two positions in $n - 2$ votes; moreover, $SG_r(P)$ loses to all but one alternative in pairwise elections. (This holds regardless of how ties are broken.)*

While this is a strong paradox already, it is sometimes possible to obtain even stronger paradoxes if we restrict attention to individual rules. We have illustrated

this on some voting rules including plurality, which can be found in Xia and Conitzer (2010b).

7.3 Computing the Backward-Induction Outcome

We have shown in the last section that the backward-induction solution to the Stackelberg voting game is socially undesirable for some profiles. We may ask ourselves whether such profiles are common, or just isolated instances that are not very likely to happen in practice. For this purpose, we would like to compare the backward-induction winner to the truthful winner by running simulations. For this purpose, we should be able to compute the backward-induction winners reasonably fast. However, even if the outcome of the rule r is easy to compute, it does not follow that the outcome of SG_r is easy to compute. The straightforward backward-induction process described above is very inefficient, because the game tree has $(m!)^n$ leaves.

In this section, we first propose a general dynamic-programming algorithm to compute $SG_r(P)$, for any anonymous voting rule r. Then, we show how to use *compilation functions* (Chevaleyre et al., 2009) (see also Section 1.6) to further reduce the time/space-complexity of the dynamic-programming algorithm. These techniques are crucial for obtaining our later experimental results.

The dynamic-programming algorithm still solves the game tree in a bottom-up fashion, but does not need to consider all the different profiles separately. Because r is anonymous, at any stage j of the game, the state (the profile of votes 1 through $j-1$) can be summarized by a vector composed of $m!$ natural numbers, one for each linear order: each number in the vector represents the number of times that the corresponding linear order appears in the $(j-1)$-profile. Formally, for any $j \leqslant n$, we let the set of these vectors (states) be $S_j = \{(s_1, \ldots, s_{m!}) \in {\mathbb{N}_{\geqslant 0}}^{m!} : \sum_{i=1}^{m!} s_i = j-1\}$. For any anonymous voting rule r and any $\vec{s} \in S_{n+1}$, let $r(\vec{s})$ be the winner for any profile that corresponds to $\vec{s}$ (because r is anonymous, the winner only depends on

the vector $\vec{s}$). More generally, for arbitrary S_j, the algorithm computes a labeling function g that maps each state $\vec{s} \in S_j$ to the alternative representing the backward-induction outcome of the subgame whose root corresponds to $\vec{s}$.

Algorithm 7.3.1.

Input. $P = (V_1, \ldots, V_n)$ and an anonymous voting rule r.

Output. $SG_r(P)$.

1. For j from $n+1$ to 1, do Step 2.

2. For any state $\vec{s} \in S_j$, do

2.1 If $j = n+1$, then let $g(\vec{s}) = r(\vec{s})$.

2.2 If $j < n+1$, then let $\vec{e}^* \in \arg\min_{\vec{e} \in E} \text{rank}(V_j, g(\vec{s}+\vec{e}))$, where E consists of all vectors that are composed of $m!-1$ zeroes and only one 1, and $\text{rank}(V_j, g(\vec{s}+\vec{e}))$ is the position of $g(\vec{s}+\vec{e})$ in V_j. (Thus, e^* corresponds to an optimal vote for j.) Then, let $g(\vec{s}) = g(\vec{s}+\vec{e}^*)$.

3. Output $g((0, \ldots, 0))$.

Analysis. For any $j \leqslant n$, $|S_j| = \binom{j+m!-2}{m!-1}$ (this is a basic combinatorial result, see e.g. Bender and Williamson (2006)). To analyze the runtime of the algorithm, we note that the total number of states considered is $\sum_{j=1}^{n+1} \binom{j+m!-2}{m!-1}$, which is $O((n+1)^{m!+1})$; in each state, we need to consider $m!$ vectors $\vec{e}$, resulting in a total bound of $O(m!(n+1)^{m!+1})$. To analyze the space requirements of the algorithm, we note that we only need to keep the last stage $j+1$ and the current stage j in memory, so that the maximum number of states in memory is $\binom{n+m!-1}{m!-1} + \binom{n+m!-2}{m!-1}$, which is $O((n+1)^{m!})$. Therefore, when m is bounded above by a constant, Algorithm 7.3.1 runs in polynomial time (using polynomial space).

However, when there is no upper bound on m, Algorithm 7.3.1 runs in exponential time and uses exponential space. We conjecture that for many common voting rules

(e.g., plurality), computing SG_r is PSPACE-hard, but we have not managed to obtain any such result yet.[4]

Compilation. In the step corresponding to stage j in Algorithm 7.3.1, a very large set S_j is used to keep track of all possible $m!$-dimensional vectors whose entries sum to exactly $j-1$, representing the possible states. While it may be necessary to have this many states for anonymous rules in general, it turns out that for specific rules like plurality or veto, we need far fewer states to represent the profiles, because many of the states in Algorithm 7.3.1 will be equivalent *for the specific rule*. For example, if we have so far received only a single vote $a \succ b \succ c$, this in general is not equivalent to having received only a single vote $a \succ c \succ b$. However, if the rule is plurality, these states are equivalent.

Pursuing this idea, for any anonymous voting rule r, we can ask the following questions. (1) *What is the smallest set of states needed for stage j?* (2) *How can we incorporate smaller sets of states into Algorithm 7.3.1?*

The answer to question (1) corresponds to the *compilation complexity* of r, a concept introduced by Chevaleyre et al. (2009). For any $k, u \in \mathbb{N}$ with $k + u = n$, the compilation complexity $C_{m,k,u}(r)$ is defined to be the smallest number of bits needed to represent all "effectively different" k-profiles, when there are u remaining votes and the winner is chosen by using r. (Two k-profiles are "effectively the same" if, for any profile of u votes that we may add to them, they result in the same outcome.) It follows that, if we tailor Algorithm 7.3.1 to a specific rule r, the size of the smallest possible set of states for stage j is between $2^{C_{m,j-1,n-j+1}(r)-1}$ and $2^{C_{m,j-1,n-j+1}(r)}$. Chevaleyre et al. (2009) also studied the compilation complexity for some common voting rules.

Now we turn to address question (2). Suppose that we have already determined

[4] We have obtained a PSPACE-hardness result for a not-so-common rule with a different type of voter preferences, which thus falls somewhat outside of the setting described so far. We omit it due to the space constraint.

that we can use a smaller set of states. In order to modify the dynamic-programming algorithm to use this smaller set of states, for step (2.2) we must have a function that takes a state in S_j and a vote V as inputs, and outputs a state in S_{j+1}; moreover, this function must be easy to compute. Fortunately, the *compilation functions* designed for some common voting rules in Chevaleyre et al. (2009) and Xia and Conitzer (2010a), which map each profile to a string (state), can serve as such functions. For example, the compilation function for plurality simply counts how often each alternative has been ranked first, and this is easy to update. More generally, we can modify Algorithm 7.3.1 for any specific rule r as follows. Let $f^r_{m,k,u}$ be a compilation function for r. For any $j \leqslant n$, we let $S_j = f^r_{m,k,u}(\mathcal{F}_{j-1})$, that is, the set of all "compressed" $(j-1)$-profiles. Then, in step (2.2), for each given state $\vec{s} \in S_j$ and each[5] given vote $V \in L(\mathcal{C})$, the next state (which lies in S_{j+1}) is computed by applying the compilation function $f^r_{m,k,u}$ to the combination of $\vec{s}$ and V. Among these resulting states, we again find voter j's most-preferred outcome.

Illustration. Let us illustrate how the use of compilation functions helps reduce the time and space requirements of Algorithm 7.3.1 for the *nomination* rule, which selects the alternative that is ranked in the first position in at least one vote, where ties are broken in the order $c_1 \succ \cdots \succ c_m$. In this case, for any $j \leqslant n$, let $S_j = \mathcal{C}$, and let f^{Nom} be the following compilation function. For any profile P, let $f^{\text{Nom}}(P)$ be the first alternative (according to the order $c_1 \succ \cdots \succ c_m$) that has been nominated (is ranked first in some vote in P). For any profile P and any vote V, $f^{\text{Nom}}(P \cup \{V\})$ can be easily computed from $f^{\text{Nom}}(P)$ and V, by determining which of $f^{\text{Nom}}(P)$ and the alternative ranked in the top position in V is earlier in the order. (As in the case of plurality, we do not need to consider every vote V: we only need to consider which alternative is ranked first.) Because $|S_j| = m$ for all j in this case, it follows

[5] For some rules, we do not need to consider every vote: for example, under plurality, we do not need to consider both $a \succ b \succ c$ and $a \succ c \succ b$.

that Algorithm 7.3.1 (using f^{Nom}) runs in polynomial time for the nomination rule.

Proposition 7.3.1. *SG_{Nom} can be computed in polynomial time (and space) by Algorithm 7.3.1 (using f^{Nom}).*

For other, more common voting rules, the runtime of the dynamic-programming algorithm is also significantly reduced by using compilation functions, though it remains exponential. For example, for plurality and veto, the time/space complexity of our approach is $O(n^m)$, which allows us to conduct the simulation experiments in the next section much more efficiently.

7.4 Experimental Results

Using the algorithmic techniques developed in the last section, we are able to run simulations to compare the backward-induction winner $SG_r(P)$ to a benchmark outcome—namely, the alternative $r(P)$ that would win under r if all voters vote truthfully. This may seem like a difficult benchmark to achieve, because often strategic behavior comes at a cost (cf. price of anarchy, first-best vs. second-best in mechanism design, etc.) Nevertheless, in the experiments that we describe in this section, it turns out that in randomly chosen profiles, in fact, slightly more voters prefer the backward-induction outcome $SG_r(P)$ to the truthful outcome $r(P)$ than vice versa!

The setup of our experiment is as follows. We study the plurality and veto rules (these are the easiest to scale to large numbers of voters, because they have low compilation complexity).[6]

For any m, n, and $r \in \{\text{Plurality}, \text{Veto}\}$, our experiment has 25,000 iterations. In each iteration, we perform the following three steps.

[6] We also investigated other rules. It appears that they may lead to similar results, though it is difficult to say this with high confidence because we can only solve for the backward-induction outcome for small numbers of voters.

1. In iteration j, an n-profile P_j is chosen uniformly at random from $\mathcal{F}_n$.

2. We calculate $SG_r(P_j)$ using Algorithm 7.3.1 (with a compilation function to reduce time/space-complexity), and we calculate $r(P_j)$.

3. We then count the number of voters in this profile P that prefer $SG_r(P)$ to $r(P)$ (according to their true preferences in P), denoted by n_1, and vice versa, denoted by n_2. If $SG_r(P) = r(P)$, then $n_1 = n_2 = 0$.

For each m, n, r, we calculate the total percentage (across all 25,000 iterations) of voters that prefer the backward-induction winner for their profile to the winner under truthful voting for their profile, that is, $p_1 = \sum_{j=1}^{25000} n_1^j/(25000n)$. We also compute $p_2 = \sum_{j=1}^{25000} n_2^j/(25000n)$. We note that it is not necessarily the case that $p_1 + p_2 = 1$, because if $SG_r(P) = r(P)$, then $n_1 = n_2 = 0$. Let $p_3 = 1 - p_1 - p_2$ be the percentage of profiles for which the backward-induction (SG_r) winner coincides with the truthful (r) winner. We are primarily interested in $p_1 - p_2$.

The results are summarized in Figure 7.1.

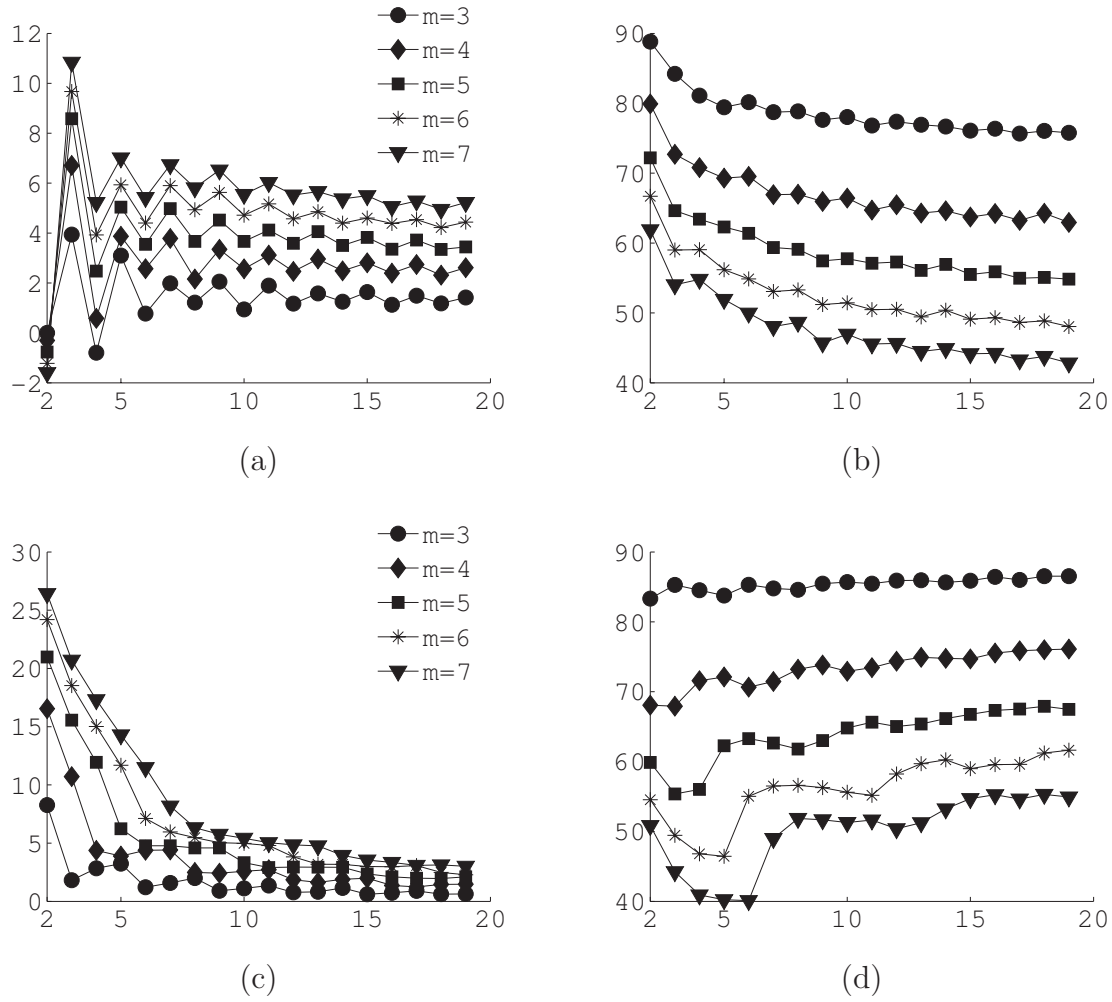

FIGURE 7.1: Simulation results for plurality and veto

In Figure 7.1, the x-axis gives the number of voters (n); the y-axis gives the percentage of voters. In each case we consider various numbers of alternatives (m). (a) The percentage of voters who prefer the SG_r winner to the r winner minus the other way around, under plurality. (b) The percentage of profiles for which the SG_r winner and the r winner are the same, under plurality. (c) The percentage of voters who prefer the SG_r winner to the r winner minus the other way around, under veto. (d) The percentage of profiles for which the SG_r winner and the truthful r winner are the same, under veto. Please note the different scales on the y-axis for (a) and

(c).

First, from (a) and (c) it can be observed that for plurality and veto, perhaps surprisingly, on average, more voters prefer the backward-induction winner to the winner under truthful voting than vice versa. Generally, the difference becomes smaller when n increases; the difference is larger when m is larger; and the percentage seems to converge to some limit as $n \to \infty$. Second, from (b) and (d) it can be observed that the percentage of profiles for which the two winners coincide is smaller for larger values of m; the percentage is decreasing in the number of voters n for plurality, but increasing for veto.

7.5 Summary

In this chapter we studied the voting game where voters cast votes one after another and the later voters can observe all previous voters' votes. We proved some paradoxes, which state that sometimes the strategic behavior of the voters can be harmful in Stackelberg voting games. To some extent, these paradoxes justify the line of research in which people seek to use computational complexity to prevent strategic behavior. We also developed algorithmic techniques to run simulations. Our simulation results show that, surprisingly, the strategic behavior of the voters does not seem as harmful as we might have expected.