Election Control: Attacking and Defending Elections using Linear Programming

Trevor Larsen and Zach Mekus
Washington University in St. Louis

Abstract

For our project, we are doing an implementation of the paper Optimal Defense Against Election Control by Deleting Vote Groups by Yin et. al. The paper introduces a double oracle approach for solving the optimization problem of how to deploy defense resources to defend an election. We are implementing this double oracle algorithm, and expanding on it by applying it to each state in the 2016 election, and propose a heuristic to decide which states to attack to optimize the probability of winning a national election using the electoral college system. Current accomplishments include implenting the Attacker-MILP and Defender-MILP functions, as well as devising the heuristic we will use for maximizing probability of success over all states in an election.

Introduction

The Problem

In recent centuries, the majority of governments in the world have shifted to forms of government that rely on elections. In order for democratic institutions to maintain their integrity, these elections must be both free and fair, as well as protected from outside interference. Malicious attackers may have motivation to influence the outcome of an election in a certain direction by subverting the democratic process. Examples of this include attacks on voting day in 2013 in Pakistan, where bombings killed or injured over 150 people, as well as in 2010 Sri Lanka elections where there were over 250 incidents of poll-related violence. Cyber attacks on electronic voter systems are also a threat, and while no known attacks have been in the United States, these may prove to be targets in the future, as recent investigations have found these systems to be vulnerable.

An attacker can attack an election with the goal of either making a certain candidate the winner, which we refer to as constructive control, or to prevent a certain candidate from winning, which we call destructive control. We are interesed in the specific example of the United States presidential election, in which we will only consider the Democratic and Republican parties. In this case, making one candidate win is equivalent to making the other candidate lose, so constructive and destructive control are equivalent. In this project, we wish to investigate how we can model an attack or defense of a national presidential election in the United States, using data from the 2016 United States presidential election.

Related Work

The first set of prior work we will look at is papers regarding election control. The Optimal Defense paper examines the problem of protecting elections against subversion by abstracting the process of enhancing security at voting locations, either physically or electronically. Prior work included Bartholdi et al. which looked at the problem from the point of view of computational (2015) were the first to look at election control at the level of voter groups, and grouped voters by a bundling function. In the paper of Yin et al. used a function where voting groups did not overlap, and where given voting groups were deleted by the advsersary (Yin et al. 2018). Erdlyi et al. added or deleted voter groups in his study, but only considered constructive control. Yin et al. in this paper was the first to consider methods for allocating defense resources for protecting given locations in an election. Elkind and Lipmaa try to make elections more difficult to subvert by developing rules that are NP-hard to manipulate, which is different from this paper, which looks at using defensive resources to reduce likelihood of an adversary changing the out come by attacking groups.

Next we will look at papers regarding physical security problems using Stackelberg games. Prior work has looked at the problem of attackers being limited to one location, whereas in Yin et al.the attackers attack multiple locations simultaneously. Other prior work has looked at simultaneous move games or have only used heuristic approaches. Other Stackelberg games have used double oracle solutions in the past, but the unique formulation of this problem requires a unique algorithm to find an optimal solution.

Current Accomplishments

We have coded and tested Core LP, AOMILP, and DOMILP, and tested it on 2016 US presidential election data for two states. We have also devised our heuristic for combining state level results to get the probability of an attacker winning the electoral college.

Model

Stackelberg Game Background

In the following section, we introduce the concept of a Stackelberg game, which we will use in the model for our election control problem. A Stackelberg game consists of two players, a leader and a follower, who each decide on a strategy for the game. The leader goes first, picking a strategy, and the follower responds with a strategy. The strategies used can either be pure strategies, where players commit to a particular action, or a mixed strategy, where players choose a probability distribution over pure strategies. A Stackelberg security game is a model for an interaction between a defending security force and an adverserial attacker. In a Stackelberg security game, because the attacker picks second, they always are able to pick the action that gives them the highest probability of success due to their knowledge of the defender's strategy, and thus will always pick a pure strategy. In contrast, the defender doesn't have any such commitment from the attacker, and usually chooses a mixed strategy to minimize the attacker's probability of success, since the defender knows the attacker will exploit any weaknesses the defender leaves in his strategy.

Let S be the set of all defender pure strategies, and A be the set of all attacker strategies. Given a defender's mixed strategy P over pure strategies in S, $P = \{p(s) : s \in S\}$, where $p(s)$ is the probability of choosing strategy s, an attacker's action $a \in A$, and utility payoff functions for the defender $U^D(s, a)$ and attacker $U^A(s, a)$ for given pure strategy (s, a) pairs we can calculate the utility of the defender and attacker:

$$u^D(P, a) = \sum_{s \in S} p(s) U^D(s, a) \tag{1}$$

$$u^A(P, a) = \sum_{s \in S} p(s) U^A(s, a) \tag{2}$$

let $br(P) \mapsto a$ be a function mapping a mixed strategy to the attacker's best response a. For a given mixed strategy, action pair (P,br(P)) to be a Strong Stackelberg Equilibrium it must fullfill the following criteria:

- $U^D(P, a) \geq U^D(P*, br(P*))$ for any other P*.
- The attacker's response $br(P) \in argmax_{a \in A} u^A(P, a)$
- defender wins ties $U^D(P, br(P)) \geq U^D(P, a), \forall a \in argmax_{a \in A} u^A(P, a)$

Model of the Election

Yin et al. model the election as a Stackelberg security game. The election cosists of people voting at different locations. At each location, each candidate receives a certain number of votes. Attacks are modeled as a deletion of the votes at these locations. In this model for the election, the defender is the leader, and first chooses a mixed strategy over m locations to defend, with the goal of defending the outcome of the election. The attacker follows with a pure strategy, choosing which n locations to attack, trying to change the result of the election, either with the goal of having a specific candidate win (constructive control) or making a particular candidate lose (destructive control). In our case, since we are modelling the US presidential election and are only considering the Republican and Democratic party candidates, we are treating the destructive and construct control cases as the same, as they become the same problem in a 2 party-election. We model each state as being an independent election composed of the districts in each state, using data compiled by Daily Kos. For simplicty, we only consider Democratic and Republican vote tallies, and use as input to the algorithm the difference in vote tallys for each district. Each state has D_n defender resources that it may allocate to defend the result of the election, while the attacker has A_n attacker resources to consider to flip the result of the election in that state. Finally, we model the national problem in the national election of having the attacker only being allowed to attack a subset of size k states.

Algorithm

In order to solve this problem, we solve a linear program which we will call Core-LP. This Core-LP takes as input the set of strategies to consider for both the attacker and the defender, as well as the vote tally difference in each district. Given these strategies and vote tallys, Core-LP computes the best defensive strategy against a following attacker, and by virtue of the dual solution, the best attacking strategy against a following defender.

Core-LP has issues with scalability, as computational time grows exponentially with the number of strategies for both the attacker and the defender. Thus, for sufficiently large elections, this linear programming approach on its own will become intractable. In their paper, Yin et al. propose a double oracle approach to solve for optimal defense and attack strategies, using heuristics to slowly grow the number of cosidered strategies. This approach involves the use of several problem-agonstic functions which must be specified for a given problem. These algorithms are the Attack-MILP,Defense-MILP,AO-Better and DO-Better functions. We cover the implementation of the functions we have implemented so far. We present the double oracle approach in Figure 1. In each iteration, the approach adds new best response pure stratgies for both the attacker and the defender to the list of considered strategies by executing AO-Better and DO-Better. If either AO-Better or DO-Better is unable to find an improved strategy, they pass off work to Attack-MILP or Defense-MILP respectively, which will return a pure strategy for the attackers and defenders to consider in the Core-LP. Once the approach is unable to find new strategies that improve upon the current set of strategies and actions, the approach terminates, returning the mixed strategy calculated by Core-LP for the defender.

Attack-MILP

In this section we describe our modified version of the Attack-MILP Linear Program that we developed for the 2-candidate scenario. In this version, we explicitly write out the linear version of the optimization problem in the original paper and removed variables that aren't relevant in a 2-candidate scenario. Attack-MILP takes as input a mixed

```
Input: S' ⊂ S; A' ⊂ A;
while do
    (x, y) ← Core-LP(S', A');
    a ← AO-Better(x);
    if a = ∅ then a ← AO-MILP(x);
    s ← DO-Better(y);
    if s = ∅ then s ← DO-MILP(y);
    if a ∈ A' and s ∈ S' then
        return x;
    else
        A' ← A' ∪ {a}, S' ← S' ∪ {s};
```

Figure 1: The double oracle algorithm proposed by Yin et al.

defense strategy generated by Core-LP. The Linear Program is given below. Let x_j be the probability of the jth defensive strategy being played. Let a_i be 1 if attack action i is taken, and 0 otherwise. Let t_i be the difference in vote count for district i. z_j is 1 if the attacker will win against pure defense strategy j. v_{ij} is a linearization variable that is 1 if district i is attacked with defense j, and will win the attack, and is calculated as $v_{ij} = a_i z_j$. S_{ij} denotes whether district i is defended in strategy j. k is the maximum number of attacks the attacker may execute.

Attack-MILP seeks to minimize the objective function:

$$min_{a_i, z_j, v_{ij} \in \{0,1\}} \sum_j (1 - z_j) x_j \tag{3}$$

subject to the following constraints:

$$\sum_i a_i \leq k \tag{4}$$

$$\sum_i (z_j t_i - (1 - S_{ij}) v_{ij} t_i) \geq 0 \forall j \tag{5}$$

$$v_{ij} \leq z_j \forall i, j \tag{6}$$
$$v_{ij} \leq a_i \forall i, j \tag{7}$$
$$v_{ij} \geq z_j + a_i - 1 \forall i, j \tag{8}$$

Defense-MILP

Let y_j be the probability of the jth attack strategy being played. Let s_i be 1 if defensize action i is taken, and 0 otherwise. Let t_i be the difference in vote count for district i. z_j is 1 if the defender will win against pure attack strategy j. v_{ij} is a linearization variable that is 1 if district i is defended with attack j, and will win the defense, and is calculated as $v_{ij} = s_i z_j$. A_{ij} denotes whether district i is attacked in strategy j. l is the maximum number of attacks the attacker may execute. Defense-MILP seeks to maximize the objective function:

$$max_{s_i, z_j, v_{ij} \in \{0,1\}} \sum_j z_j y_j \tag{9}$$

subject to the following constraints:

$$\sum_i s_i \leq l \tag{10}$$

$$\sum_i (z_j t_i (1 - A_{ij}) + v_{ij} t_i A_{ij} + z_j \leq 0 \forall j \tag{11}$$

$$v_{ij} \leq z_j \forall i, j \tag{12}$$
$$v_{ij} \leq s_i \forall i, j \tag{13}$$
$$v_{ij} \geq z_j + s_i - 1 \forall i, j \tag{14}$$

Michigan: Clinton Voters - Trump Voters by District (Thousands)

District	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Vote Difference	-78	-61	-33	-81	14	-28	-58	-26	27	-115	-17	89	161	194

3 defenders, 2 attackers

Michigan: Probability of each District Being Selected to Attack or Defend

District	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Defense (Primal)	0.33	0.33	0.33	0.33	0	0.33	0.33	0.33	0	0.33	0.33	0	0	0
Attack (Dual)	0.22	0.22	0.22	0.22	0	0.22	0.22	0.22	0	0.22	0.22	0	0	0

Objective (Probability of successful defense): .083

National Election

Like the AO-Better and DO-Better functions, we have yet to implement the National Election heuristic for our project, but we already have our heuristic for choosing states in mind. Recall that using the double oracle approach we are able to calculate a probability of flipping each state to a given party. In the U.S. presidential election the goal is to win the majority, which is 270, of the electoral college votes. Each state is assigned a number of electoral college votes in each election. Our heuristic is straightforward. If the attacker wants to win the election for a particular candidate, he should maximize the expected number of votes that candidate will achieve. Thus, our heuristic is to take the number of electoral votes EV_n in each state, and multiply this by the probability of winning the state given by the double oracle approach in that state PW(n).

$$E[EV] = EV_n * PW(n) \tag{15}$$

We then sort the states by E[EV] and select the k largest values. We will include the results of this approach in our final report.

Results/Evaluation

Overview

The results section has the following goals: To show that what has been implemented so far is correct, to get an initial evaluation of our states data, and to gage the limitations of Core LP in terms of run time on further state data. We

Michigan: Results of Attacker and Defender Oracles with input from Core LP

	Decision	Objective Value
AOMILP	Attack 3, 7	0.083
DOMILP	Defend 3, 5, 6	0.083

Michigan: Performance of Optimizations

	Continous Variables	Binary Variables	Constraints	Time (s)
Core LP	365		92	0.59
AOMILP		5474	15653	2.25
DOMILP		1379	3914	0.64

Georgia: Clinton Voters - Trump Voters by District (Thousands)

District	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Vote Difference	-42	29	-98	158	223	-5	-18	-77	-174	-80	-82	-43	134	-135

3 defenders, 5 attackers

Georgia: Probability of each District Being Selected to Attack or Defend

District	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Defense (Primal)	0	0	0.42	0	0	0	0	0.32	0.71	0.42	0.42	0	0	0.71
Attack (Dual)	0.40	0	0.66	0	0	0	0.21	0.79	0.55	0.79	0.79	0.19	0	0.61

Objective (Probability of successful defense): .29

Georgia: Results of Attacker and Defender Oracles with input from Core LP

	Decision	Objective Value
AOMILP	Attack 2, 7, 9, 10, 13	0.29
DOMILP	Defend 2, 8 13	0.29

Georgia: Performance of Optimizations

	Continous Variables	Binary Variables	Constraints	Time (s)
Core LP	365		2003	10.03
AOMILP		5474	15653	2.40
DOMILP		30044	86078	14.99

evaluate on 2016 US presidential election data using the difference in votes between Hillary Clinton and Donald Trump each congressional district of the states Michigan and Georgia. The optimal randomized defensive strategy is solved using Core LP and the dual solution of Core LP is the optimal randomized attacking strategy for the scenario where the attacker moves first and the defender plays a pure strategy. AOMILP and DOMILP work by taking randomized strategies and returning the optimal attacking and defending strategy respectively. To ensure these will work later on in the double oracle, these are tested here to make sure they give correct strategies and objective values. The probability vectors over the districts are calculated by finding the expectation that each district will be defended or attacked based on the randomized strategy.

Michigan

In 2016 the presidential election results went narrowly to Donald Trump. If any district that favored Trump is deleted, the results would have changed. Here we use three defenders and two attackers, but the results would have been similar no matter what. Since every district is equally valuable, the defenders and attackers pretect them equally. Table shows the equal distribution among districts the defender won, in both the attacking and defending solutions (defending sums to 3, attacking sums to 2). This is also seen in the response in table by the attacker and defender, which arbitrarily choose districts among those the defender won.

Georgia

While Georgia was competitive, it wasn't so close that any To induce a more interesting result, 3 defenders and 5 attackers were introduced and this resulted in a 29% chance of defending successfully. As seen in table the larger defender-won districts got more defense and attack. Because the attackers had more options, they could sometimes attack districts never defended. Table shows that each of the oracles corresponds to choosing the most attacked/defended districts, but not neccessarily the ones with the most votes to destroy.

Results Conclusion

The two oracles AOMILP and DOMILP appear to have performed correctly, especially by getting the same optimal value as Core LP. As expected, with more defender and attacker strategies, the optimizations of Georgia took longer to compute. This certainly seems to be a limitation on the future of using Core LP on larger states or using lower level districts like counties. Our next step will be replicating the double oracle algorithm to avoid this.

Related Work

We experiment on the same data as Yin et al. with 2016 Michigan presidential data. With the same settings, three defense and two attack resources, we both get the same result of .29 success rate. This is promising as it serves as further confirmation that we have implemented Core LP corectly. In the Results section, we expand on the choices made by

the defensive and attacking agents in Michigan and Georgia to further understand the process. In the future, we hope to expand on the solving this problem for all states.

Conclusion, Limitations, and Future Work

In summary, our project is well on the way to completion. Our Implementations of the Core-LP, Attack-MILP and Defense-MILP are working as expected. One limitation of the approach we have taken is that runtime can still be prohibitive for states with many districts. Additionally districts are a poor representation of voting units since a district has many counties within it, and counties have many voting centers within them. Analyzing group voting at the level of voting centers is far more realistic, though computationally much more porhibitive. Additionally, it is difficult to find voting data at the level of individual voting centers. Counties are also too numerous to be computationally tractable for a project with our computational resources. Future work for this project will include finishing the implementation of the double-oracle approach by completing AO-Better and DO-Better, and implementing our heuristic at the national election level. Finally, we also need to compare the results of our finished work to those produced by Yin et al.

References

Yin, Y.; Vorobeychik, Y.; An, B.; and Hazon, N. 2018. Optimal defense against election control by deleting voter groups. *Artificial Intelligence* 259.