Budget and Time Distribution Optimization to Win an Election

Operational Research II

Fall 2020

Elena Chen

Ruiqi Wang

Yue Wu

Leonard Zhang

Outline

Introduction

Executive Summary

In the 2020 presidential election, nearly $14 billion was raised and spent to support the candidates during the presidential campaign. The sum is more than double of what was spent in the 2016 election. A major amount of the raised fund was spent on the election campaigns in every state. But had the candidates really spent the money in need? This paper discusses our efforts to provide solutions for election candidates to allocate their money and time reasonably to gain as many electoral votes as possible.

Project Objective

In this project, we explore the voting problem of two parties over several time periods, a sequential move game. Both parties aim to get the majority electoral vote by the end of the final period in order to win the election. Therefore, they devise strategies to invest time and endowment for states to gain votes. Their actions are based on their budget constraint and the other party's strategy. We provide algorithms and strategies that potentially increase a certain party's chances of winning.

The Voting Problem

In the 2020 Pineapple country's presidential election, there are two parties running for the presidency: the Pine Party and the Apple Party.

There are n states in Pineapple country. Each state has the number of citizens $S = \{s_1, s_2, \ldots, s_n\}$, every citizen votes for one party. A state's electoral vote is proportional to its number of citizens $E = \{e_1, e_2, \ldots, e_n\}$, where $e_i = s_i \,//\, k$, (factor by k and round down to a whole number). All of the electoral votes of a state are given to the candidate that gets the popular vote.

At each timestamp $t = j \in \{0, 1, ..., T\}$, each state $i \in \{0, 1, ..., n\}$ has a preference for a party:

(number of votes for Pine, number of votes for Apple) = $\left(p_i^{(j)}, a_i^{(j)}\right)$, $p_i^{(j)} + a_i^{(j)} = s_i$

*** the superscripts for time are omitted in the rest of this paper**

At each time period t to $t+1$, the parties want to alter the preferences so that they could win the majority of electoral votes at the terminal timestamp, $t = T$. To do so, they spend time and money on each state. The parties take sequential moves, meaning that if the Apple Party moves at every odd time period (from $t = 2m$ to $t = 2m+1$, $m \in \mathbb{Z}^+ \cup \{0\}$), then the Pine Party would move at every even time period. For both parties, the cost of altering one popular vote in state i is $w_i \in [0, 1]$ dollars of endowment and $d_i > 0$ amount of time.

At initial $t = 0$, each candidate is given W dollars of endowment, and there won't be other sources of funding throughout time. Each candidate also has 1 unit of time at each time period from t to $t+1$.

You are the strategist of the **Apple Party**. You know that the Pine Party has a competitive strategist and you know his/her strategy. Your job is to come up with a best-for-winning strategy, for making the best move in spending endowment and allocating time, or trapping your opponent, based on Pine Party's best strategy.

Assumptions & Set Up

Before we start building our mathematical models, we would like to emphasize on several significant assumptions made in our problem statement:

1. Everyone in every state will vote. That means there will be total numbers of $S = \{s_1, s_2, \ldots, s_n\}$ votes in every single state.
2. Every move is made sequentially rather than simultaneously. That is, Apple Party and Pine Party will each move sequentially. The assumption is valid because if we take every interval small, the sequential game reflects real-world voting scenario, when one party responds to the other party's move in a few hours or days.

Based on these assumptions, we are able to refine our problem - **for each party, at each of its steps, aims at deriving its maximum utility obtained from its actions.** The

utility functions $U_{ij},\ i \in \{1,\ 2,\ \ldots n\},\ j \in \{a,\ p\}$ of one vote at every state i are specifically defined as below.

Utility per vote at state i:

$$\text{Apple: } U_{ia}(W_a, d_i, w_i, e_i, s_i, a_i) = \alpha_a \times \frac{1}{W_a \times w_i} + \beta_a \times \frac{1}{d_i} + \gamma_a \times \frac{e_i}{\frac{s_i}{2} + 1 - a_i}$$

$$\text{Pine: } U_{ip}(W_p, d_i, w_i, e_i, s_i, p_i) = \alpha_p \times \frac{1}{W_p \times w_i} + \beta_p \times \frac{1}{d_i} + \gamma_p \times \frac{e_i}{\frac{s_i}{2} + 1 - p_i}$$

where $\alpha_j, \beta_j, \gamma_j, j \in \{a, p\}$ are preference constants.

Recall that:

$W_{a,}\ W_p$: remaining dollars of endowment for each party;

d_i : time cost to win over one vote at state i;

w_i : dollar cost to win over one vote at state i;

e_i : total electoral vote of state i;

s_i : number of popular votes, or number of citizens at state i;

$a_i,\ p_i$: current number of votes won by Apple and Pine party respectively.

- The utility function is negatively associated with
    - Remaining endowment: having less endowment adds more weight to the budget term $1 / w_i$, meaning that the utility is more sensitive to monetary cost
    - Time cost for one vote: utility decreases with cost
    - Money cost for one vote: utility decreases with cost
    - Current vote gap: preference for smaller vote gaps, easier to get tied
- The utility function is positively associated with
    - Number of electoral votes

We also divide our discussions into the following steps:

Problem I

Problem Statement

We first proposed that **each party will allocate its resources in order to win the most "valuable" state at every single time step**. "Winning" a state refers to obtaining more than half of the total votes from that state. This could be achieved by greedily solving a knapsack problem.

We set a simple objective function for both parties: try to maximize the total utility that we could get during every single time period with a fixed amount of time and endowment constraint. At every time step, each of the party has n items with value $U_{ij} \times votesNeeded,\ i \in \{1,\ 2,\ ..., n\},\ j \in \{a,\ p\}$, where $votesNeeded$ is the votes required to win more than half of the total votes at state i. To choose item i it costs $w_i \in [0, 1]$ dollars of endowment and $d_i > 0$ amount of time.

Mathematically, at each of their time periods, each party solves for the number of votes added for each state $x = [x_1, ..., x_n]$:

Apple:

$$\max_{\vec{x}} U_a = \sum_{i=1}^{n} utility_a(W_a, d_i, w_i, e_i, s_i, p_i) \times x_i$$

$$sub.to \sum_{i=1}^{n} w_i x_i \le W_a$$

$$\sum_{i=1}^{n} d_i x_i \le 1$$

$$0 \le x_i \le s_i - a_i$$

$$w_i, d_i, x_i \ge 0, \forall i \in [1, n] \cap \mathbb{Z}$$

Pine:

$$\max_{\vec{x}} U_p = \sum_{i=1}^{n} utility_p(W_p, d_i, w_i, e_i, s_i, a_i) \times x_i$$

$$sub.to \sum_{i=1}^{n} w_i x_i \le W_p$$

$$\sum_{i=1}^{n} d_i x_i \le 1$$

$$0 \le x_i \le s_i - p_i$$

$$w_i, d_i \ge 0, \forall i \in [1, n] \cap \mathbb{Z}$$

Apple: $U_{ia}(W_a, d_i, w_i, e_i, s_i, a_i) = \alpha \times \frac{1}{W_a \times w_i} + \beta \times \frac{1}{d_i} + \gamma \times \frac{e_i}{\frac{s_i}{2} + 1 - a_i}$

Pine: $U_{ip}(W_p, d_i, w_i, e_i, s_i, p_i) = \alpha \times \frac{1}{W_p \times w_i} + \beta \times \frac{1}{d_i} + \gamma \times \frac{e_i}{\frac{s_i}{2} + 1 - p_i}$

where α, β, γ are preference constants. Here, we assume they are equal for both parties.

where n denotes the total number of states, W denotes the amount of available budget at that period (remaining endowment), D denotes the amount of available time, w_i denotes the amount of money we spent in state i, d_i denotes the amount of time we spent in state i and function $Utility_{ij}(W_{j,} d_{i,} w_{i,} e_{i,} s_{i,} j_i), j \in \{a, p\}$ returns the utility that we could get by spending w_i amount of money and d_i amount of time in state i.

To simplify the problem, we assume that initially, each party received an equal amount of endowment, and they get the same units of time at each interval. Also, we assume both of them adopt the same strategy of maximizing the utility earned in one single interval. Also, for each party, they would try to retrieve a defeat in the states that they are currently falling behind.

Solution I

We can formulate the problem by first getting all the possible strategies s = [strategy_1, strategy_2, ..., strategy_t] that could be adopted in that single period where each strategy aims at winning one more popular vote over the opponent. Then, we sort these strategies according to their utility, and find the set of strategies that generate the maximum amount of total utility while meeting the budget and time constraint. Then, we update the current state according to the strategies that we have just determined, and step forward to the next time interval. For each party, we defined their utility function in the form that we described above. At the very end of the simulation, we could get a list of strategies that the two parties adapted across time and the final winner of the election.

Results of Solution I

After adapting our first solution, we run the whole algorithm on different initial state for a few times, and we get the following results:

First, we run 30 trials under the situation that the we (Apple Party) have an initial lead in the election of 10 states. Both parties start with an endowment of 10,000 and we run our simulation algorithm for 20 time periods. Among the 30 trials, at the end of the election, the Apple Party managed to win 2 of them.

Sample Trial: 10 states, 20 time periods, each party with endowment 10,000:
First Mover: Apple Last Mover: Pine

Timestamps	Apple's Electoral Votes	Pine's Electoral Votes	Winner
Initial t = 0	115	65	Apple Party
Final T = 20	70	110	Pine Party

Next, we run 30 trials under the situation that our opponent (Pine Party) has an initial lead in the election of 10 states. Both parties start with an endowment of 10,000 and we run our simulation algorithm for 20 time periods. Among the 30 trials, at the end of the election, the Pine Party won every single election.

Sample Trial: 10 states, 20 time periods, each party with endowment 10,000:
First Mover: Apple Last Mover: Pine

Timestamps	Apple's Electoral Votes	Pine's Electoral Votes	Winner
Initial t = 0	67	73	Pine Party
Final T = 20	13	127	Pine Party

Finally, we run 30 trials under the situation that the we (Apple Party) have an initial lead in the election of 20 states. Both parties start with an endowment of 10,000 and we run our simulation algorithm for 20 time periods. Among the 30 trials, at the end of the election, the Apple Party managed to win 5 of them.

Sample Trial: 20 states, 20 time periods, each party with endowment 10,000:
First Mover: Apple Last Mover: Pine

Timestamps	Apple's Electoral Votes	Pine's Electoral Votes	Winner
Initial t = 0	175	140	Apple Party
Final T = 20	109	206	Pine Party

From running our simulation algorithm under three different kinds of settings, we could observe that compared to the first mover, **the last mover owns a significant advantage**. If the last mover had the initial lead, it could hardly lose any election. However, even if the first mover had a significant lead at t = 0, it is very likely that the last mover could take a turn and win the final election.

This trend suggests that our model is more beneficial to the last mover because the greedy strategy that both parties adopt is to win by one vote in each state and revert as many states as possible. Under such a setting, it is obvious that the last mover is more likely to win the battleground states.

Problem II: Last-Mover Benefit

Problem Statement

After running several trials, we realize that the assumptions we made **significantly benefit the last mover**, who has the ability to win over states that were won by the opponent in the second last time period. Therefore, we decided to solve the problem by **inducing backwards**.

Similarly to Solution I, we assume that the last mover uses the utility function to solve the knapsack problem: to "flip" as many states that are worth the most electoral votes as possible.

Acting as the second last mover, we will then aim at minimizing the maximum utility function that could be achieved by the last mover. Here, we want to come up with a strategy for the second to last mover such that it can reduce the advantage of the last mover as much as possible.

Solution II

Before describing the second solution, we will first define two functions: **Update** and **MaxPineUtility**. Function Update will take in list of votes for Apple Party A = [$a_1, a_2, ..., a_n$], list of votes for Pine Party P = [$p_1, p_2, ..., p_n$] and an action list X = [$x_1, x_2, ..., x_n$] and update A and P accordingly.

Mathematically, at the last time period, the last mover solves,

Last Mover - Pine:

$$\max_{\vec{x}=[x_1,\ldots,x_n]} Utility_p\left(W_p, \vec{d}, \vec{w}, \vec{e}, \vec{a}, \vec{s}, \vec{p}, \vec{x}\right) = \sum_{i=1}^{n} U_{ip}\left(W_p, d_i, w_i, e_i, s_i, a_i\right) \times x_i$$

$$sub.to \sum_{i=1}^{n} w_i x_i \leq W_p$$

$$\sum_{i=1}^{n} d_i x_i \leq 1$$

$$0 \leq x_i \leq s_i - p_i$$

$$w_i, d_i \geq 0, \forall i \in [1, n] \cap \mathbb{Z}$$

Where $U_{ip}\left(W_p, d_i, w_i, e_i, s_i, p_i\right) = \alpha_p \times \frac{1}{W_p \times w_i} + \beta_p \times \frac{1}{d_i} + \gamma_p \times \frac{e_i}{\frac{s_i}{2}+1-p_i}$

The second to last mover solves,

Second to Last Mover - Apple:

$$\max_{\vec{y}=[y_1,\ldots,y_n]} Update_a\left(W_a, \vec{d}, \vec{w}, \vec{e}, \vec{a}, \vec{s}, \vec{p}, \vec{y}\right)$$
$$= \sum_{i=1}^{n} e_i \left(\mathbb{I}\left\{s_i + y_i > \frac{s_i}{2}\right\} - \mathbb{I}\left\{s_i - y_i < \frac{s_i}{2}\right\}\right)$$
$$- \max_{\vec{x}=[x_1,\ldots,x_n]} Utility_p\left(W_p, \vec{d}, \vec{w}, \vec{e}, \vec{a} + \vec{y}, \vec{s}, \vec{p} - \vec{y}, \vec{x}\right)$$

$$sub.to \sum_{i=1}^{n} w_i y_i \leq W_a$$

$$\sum_{i=1}^{n} d_i\, y_i \leq 1$$

$$0 \leq y_i \leq s_i - a_i$$

$$w_i, d_i \geq 0, \forall i \in [1, n] \cap \mathbb{Z}$$

Where in the objective function, we minimize the $Utility_p$ function defined for the last mover

Results of Solution II

We realize that **the objective functions of Problem II are not linear**. Inducing backward using Game Theory would simply mean we have multiple potential strategies at every time step, yielding millions of total strategies to evaluate at and choosing the $argmax_{strategy} U$ from. This is apparently challenging at this stage and will be left as a topic for extended research and study in the near future.

Problem III: Tiny Winning Margin

Problem Statement

It then came to our concern that we should not be expecting two parties using the same strategies. Meanwhile, the assumption that each party, at every time step, aims at only winning a state i by one popular vote, is also rather naive. Situations would be extremely volatile, as subsequent moves for the counterparty would be heavily focused on winning over 2 votes at state i, to gain a minor advantage. Additionally, we ought to decide the optimal ways to spend our endowment at every time period. Hence, instead of fixing our initial endowment as a constant, we increase both parties' endowment at every time period, allowing more efficient use of dollars to maximize utility functions.

Solution III

For our third solution, we improve our Solution I and II in the following aspect:

- For each party, their target is now achieving a lead of at least 10% in each state instead of the original naive aim of winning a state by one single popular vote. This assumption can help the first mover reduce the advantage of the last mover since it will be increasingly harder for the last mover to revert the opponent's lead in one state in a single interval. Also, this assumption is more consistent with the reality because in our Solution I, even if one party only leads by one single vote in a certain state, it will not consider consolidating its lead. Now, in Solution III, the party will take this option into consideration.

- While we still assume that the last mover uses the utility function to solve the knapsack problem: to "flip" as many states that are worth the most electoral votes as possible, we give the first mover a different strategy: always prioritize the investment in states with lower amortized cost per vote. This means that we will keep track of the number of electoral votes gained per dollar spent, and would first invest in the states that could bring us more electoral votes with lower cost. To implement this feature, we sort all the states by (electoral vote / cost to

flip), and solve a knapsack problem to find the optimal strategy set to be taken in one single time period.

- Finally, we improve our implementation to study the optimal budget distribution overtime. In Solution I, every party starts by holding all endowment and spends as much endowment as possible to gain a lead at the early stage of the election. In order to discover an optimal distribution of endowment, we update our implementation to allow the two parties to choose different kinds of budget distribution strategies: uniform, invest more endowment at the last few time intervals, and invest more endowment at the early stage of the election. By comparing the winning rate under the same setting for all three strategies, we can find the optimal one.

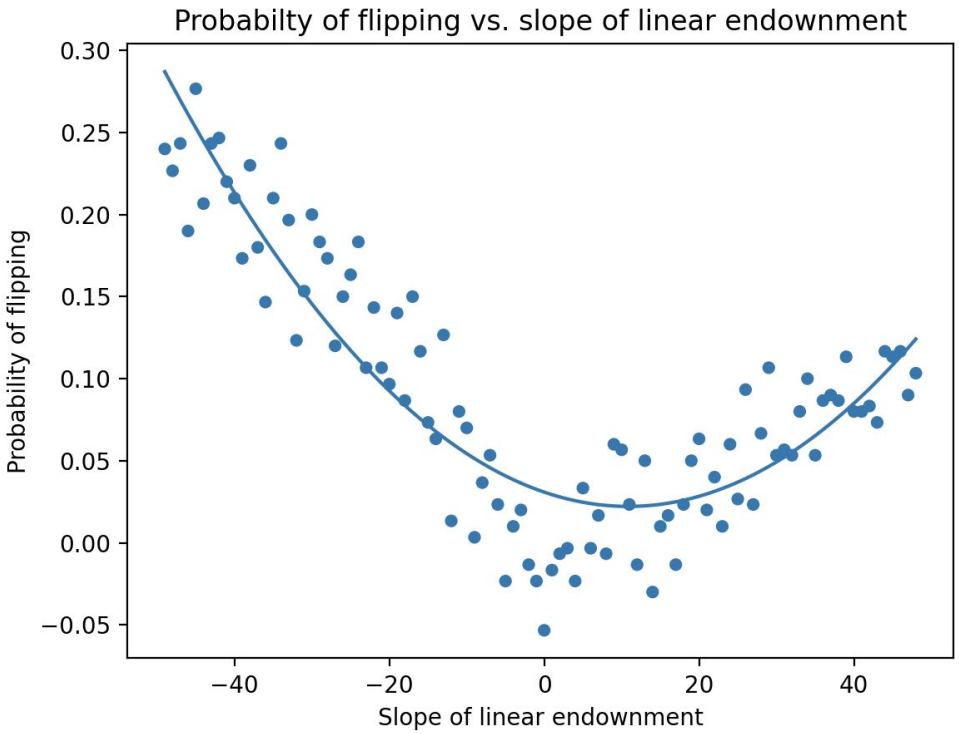
Results of Solution III

In this section, we first analyze the performance of Solution III, from the perspective of endowment allocation, number of endowments, and initial states. Then, comparing Solution I and Solution III, we analyze their performance in terms of the chance of flipping the result for the us (Apple party).

- **Endowment allocation - a linear approach**
  With all other variables fixed, we allocate the endowment as a linear function of the time. For example, with a total endowment of 10000 and 20 time periods, such an allocation with slope 50 can be represented as [0, 50, 100, ..., 1000]. We defined probability of flipping as the chance that the Apple party won the election even if its initial situation was a loss. As the Apple party, we would like to maximize this probability. The following plot represents how probability of flipping changes with respect to the slope of linear endowment allocation. Using a second degree polynomial regression, we have:
  $P(flipping) = 7.394e-05\, k^2 - 0.001606\, k + 0.03086$.

  From both the regression and the plot, we can see that we have a high probability of flipping when the slope is negative. That is to say, the Apple party has an

advantage when it spends most of the endowments at the beginning of the election.



- **Endowment allocation - discussing concavity**

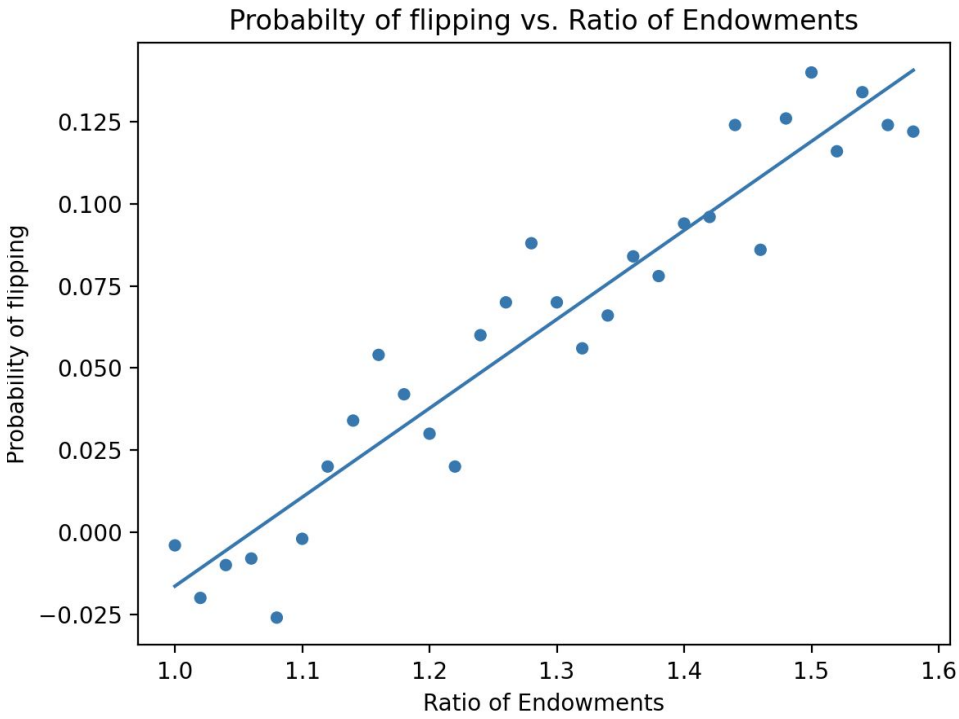
  In the last section, we already figured out that decreasing allocations of endowments works better than increasing ones in most cases. In this section, we would like to release the constraint of linearity. Specifically, we want to figure out whether a concave up, decreasing allocation or a concave down, decreasing allocation works better. We choose both the concave up or down function as a second degree polynomial with respect to the index of time. For each scenario, we run 100,000 trails, and have the following result:

	Probability of Flipping
Concave down	5.174%
Concave up	8.042%

From the table we find that the concave up decreasing function performs better. Notice that this finding aligns with the previous finding that “the Apple party has an advantage when it spends most of the endowments at the beginning of the election.”

- **Total Endowments**

  In this section, we explore how extra endowments change the election result. This problem is essential as it answers whether the party should spend time raising more money for its campaign. We define the ratio of endowments as (endowments of the Apple party)/(endowments of the Pine party). Also, we define the probability of flipping the same as in the last section. We find a roughly linear correlation between these two variables.



- **Initial States**

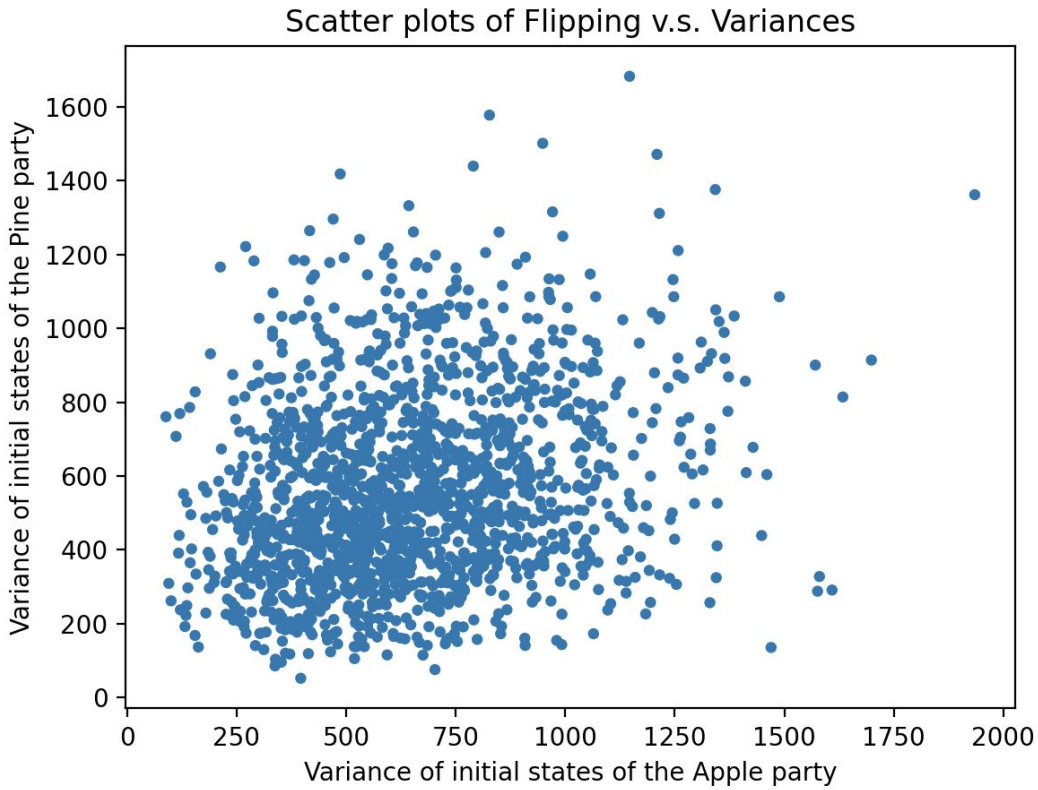
  In this section, we try to figure out under what conditions can the Apple party win. The question is essential as we might want a prediction of the results based on the initial situations.

  - **Variance of initial states**

Intuitively, the variance of the initial states impacts how hard it is to flip the result. That is, if we have a high variance, we probably need to spend many efforts on several specific states to make them flip; if we have a low variance, we need to make efforts on a larger number of states. We would like to figure out how the algorithm performs under both situations.

Define the variance of the initial states of the Apple/Pine party as the variance of the initial vector that represents Apple's/Pine's initial votes. Then, run our randomized algorithms for 30,000 times, and filter the trails when we have a flip, that is, the Apple party turns a loss into a win. The scatter plot is as follows. We can see that it is hard to make a flip when the variance is high.

Admittedly, there are possible flaws with this analysis. Since the total votes are random, a high variance might correlate to a high total votes. Since the number of endowments is fixed, it is much harder to flip the result under such a situation. Such flaws will be eliminated in the next section, where we quantify both variance and the initial advantage that the Apple party has.

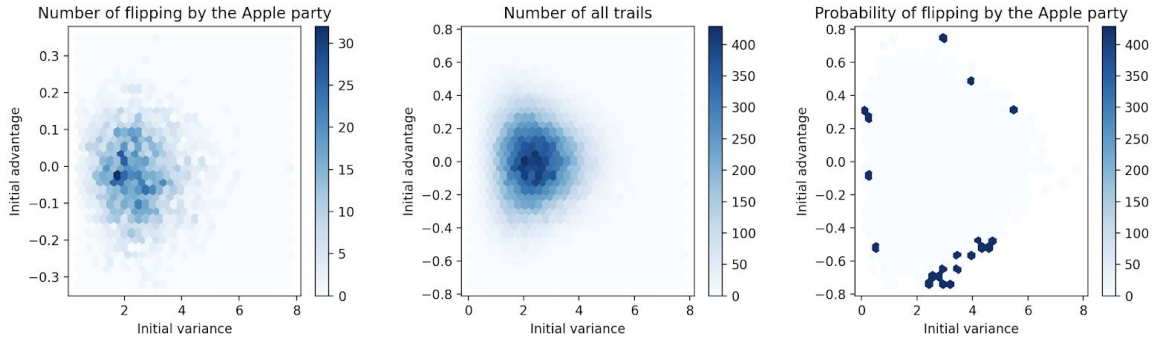


- **Variance and Initial Advantage**

  Intuitively, beyond variance, the initial advantage also contributes to the ease with which a party can win. In this section, we are going to quantify both the variance and the initial advantage.

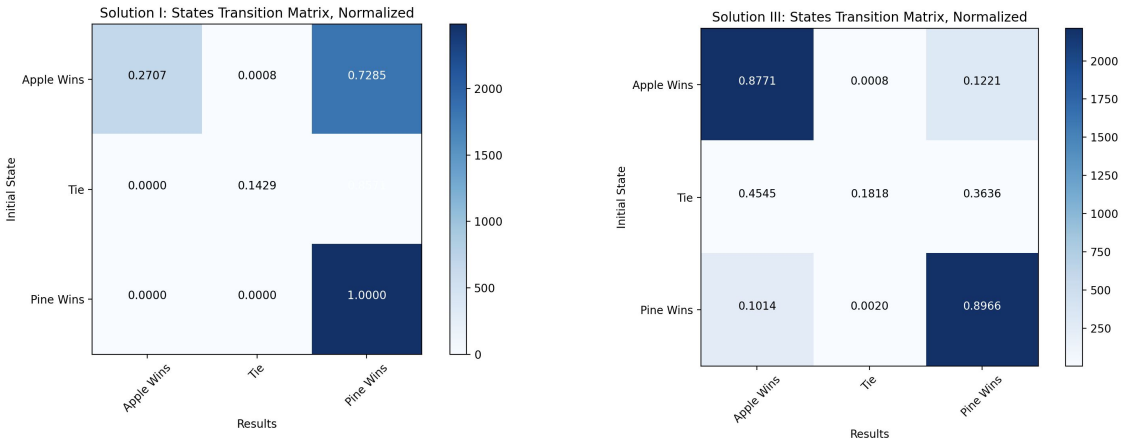
  To eliminate the impact of randomized state votes, we define both the initial variance and the initial advantage in a relative way. Let A and P represent the initial votes of the Apple party and the Pine party. Let S be the array of total votes. Note that S[i] = A[i] + P[i] for all i. We define the relative variance here to be variance(A-P)/sum(S), and define the relative advantage to be (A-P)/sum(S). Then, we run 50,000 trails and use The left plot shows the trails where the Apple party flips the result. The middle one plots the distribution of all trails. The right plot is defined to be the ratio of the left and middle, and thus representing the probability of flipping by the Apple party.

From the following plot, especially the right one, we can conclude that the probability of flipping is roughly uniform when the initial advantage and variance are not extreme, as we can see light blue spread uniformly across the middle of the right graph. One interesting finding is that this algorithm allows the Apple party to flip with a good probability under extreme initial conditions, as we can see dark blue grids spreading around the middle light part.



Conclusion

- **Comparing Solution I and Solution III:**
  To compare the results of Solution I and Solution III, we respectively generate the following probability transition matrix from 5000 trails to represent the result:



  As we can conclude from the above probability transition matrix, solution III works better, as it can sometimes prevent the pine party from winning. Furthermore, from the results that we got from Solution III, we knew that the Apple party has an advantage when it spends most of the endowments at the beginning stage of the election. Hence, the first mover should adopt the strategy of prioritizing the investment in states with lower amortized cost per vote and spending more endowment at the beginning of the election to establish a solid advantage over the opponent.

- **Limitations and Potential Extensions of Study:**
  In this study, we made a naive assumption on the opponent's strategy to simplify the implementation progress. In our implementation, our opponent will always try to maximize its utility greedily in one single period. One potential extension of our study is to design a few kinds of different strategies in advance and let both parties take different sets of strategies. Then, we could better compare different strategies and conclude the advantages and disadvantages of each strategy.

Also, in our study, one party will stick to one single strategy during a trial, but in reality, changing strategy according to your opponent's move is necessary. Thus, we could design mixed strategies for both parties and compare their payoff with pure strategies to find a better optimization.