Bribery over Social Networks

BTP Project-I report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Abhinav Sen

(20CS10003)

Under the supervision of

Professor Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2016-17

November 04, 2023

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November 04, 2023 (Abhinav Sen)

Place: Kharagpur (20CS10003)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Bribery over Social Networks**" submitted by **Abhinav Sen** (Roll No. 20CS10003) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2016-17.

3-11-23

Professor Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Date: November 04, 2023

Place: Kharagpur

Abstract

Name of the student: **Abhinav Sen** Roll No: **20CS10003**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Engineering**

Thesis title: **Bribery over Social Networks**

Thesis supervisor: **Professor Palash Dey**

Month and year of thesis submission: **November 04, 2023**

Bribery has long been recognized as a pervasive issue in the context of elections and decision-making processes. This thesis delves into the realm of bribery over social networks, with a specific focus on understanding and optimizing resource allocation in an environment where voters are interconnected, represented as a graph. Each voter in this network possesses an associated cost function, defining the expense required to bribe them. We then look for algorithms to find optimal bribery schemes under a predefined budget to successfully change the course of the election. We do this for special cases and constraints as well to study tractable solutions.

This research contributes to the understanding of resource allocation under bribery schemes in interconnected social networks. By addressing the complexity and algorithmic aspects of the problem, we offer insights into mitigating the influence of bribery in decision-making processes while considering the intricacies of social network structures. The findings of this thesis can potentially inform policy decisions, election strategies, and algorithmic solutions to combat bribery within the modern world.

Acknowledgements

We would like to express our sincere gratitude to the following individuals, institutions, and organizations whose contributions and support were instrumental in the completion of this research paper.

First and foremost, we extend our deepest appreciation to our research supervisor, Palash Dey, whose guidance, expertise, and unwavering support have been invaluable throughout the entire research process.

Our heartfelt thanks go to the staff and facilities at IIT Kharagpur for providing the necessary resources, library access, and research assistance. Your institution's support played a pivotal role in shaping this research.

Abhinav Sen

Contents

Chapter 1

Motivation

1.1 Introduction

While previous works exist on control and bribery in voting, none of them consider the inter-relationships of the actors being manipulated. This makes them highly detached from real life, where inter-relationships are highly important in manipulation strategies.

1.2 Literature Survey

A major collection of results in contemporary computational social choice can be found in F. Brandt and Procaccia (2015). A lot of important its results can be derived from here as well (Bartholdi et al., 1989) and (Bartholdi III et al., 1992), which were the precursors to the study of social choice computationally. The papers talk about the complexity of various bribery schemes under various voting rules. For the purpose of this thesis, we mostly look at the majority and plurality voting rule, which always select the condorset winner ((F. Brandt and Procaccia, 2015)), if it exists. From (Bartholdi III et al., 1992) we can see that WEIGHTED SHIFT BRIBERY problem is $\mathcal{NP}$-complete for the plurality rule, while the SHIFT BRIBERY with plurality is polynomial solvable. From (Bredereck et al., 2016) we

can see that Combinatorial Shift Bribery problem is $\mathcal{NP}$-complete and W[1]-hard even for the Plurality rule for only five voters and two candidates and no budget constraints.

Clearly, the class or problems that arise from social choice theory are often non trivial in nature. We attempt a more generalised version of the problems studied so far.

1.3 Research gaps

While the computational complexity of various bribery schemes under various voting rules have been studied, none of these studies have taken into account the interactions between the voters being bribed. For example, it might be sensible to bribe a voter with high influence, who can also impact other voters. We model the influence of one voter on others through a weighted graph of influences. This produces a more holistic and general model to the bribery problem that is more closely tied to real life.

1.4 Objective

The primary objective of this project is to explore the complexity and develop algorithms to identify optimal resource allocation strategies under a bribery scheme within interconnected social networks. By leveraging the relationships between voters as represented by the graph structure, we aim to answer critical questions pertaining to the efficiency, fairness, and ethical implications of such resource allocation schemes.

The thesis provides a rigorous analysis of the computational complexity of this problem. We investigate the fundamental questions of whether the problem is NP-hard, and if so, we aim to determine the boundaries of its complexity. We also explore special cases and constraints that can lead to tractable solutions.

Chapter 2

Preliminaries

2.1 Introduction

Let us look at some commonly used terminology in the study of control and bribery

2.2 Commonly used terminology

An election is a pair $(\mathcal{C}, \mathcal{V})$, where $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of candidates and $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a set of voters. If not mentioned otherwise, we use n and m to respectively denote the number of voters and candidates. Each voter v_i has a preference order (vote) $\succ_i$, which is a linear order over $\mathcal{C}$. We denote the set of all complete orders over $\mathcal{C}$ by $\mathcal{L}(\mathcal{C})$. We call a list of n preference orders $\{\succ_1, \succ_2, \ldots, \succ_n\} \in \mathcal{L}(\mathcal{C})^n$ an n-voter preference profile. We denote the i^{th} preference order of a preference profile $\mathcal{P}$ as $\succ_i^{\mathcal{P}}$. A map $r : \mathcal{L}(\mathcal{C})^n \to \mathcal{C}$ is called a resolute voting rule (as we assume the unique-winner model); in case of ties, the winner is decided by a *lexicographic* tie-breaking mechanism $\succ_t$, which is some pre-fixed ordering over the candidates. For a set of candidates X, let $\overrightarrow{X}$ be an ordering over them. Then, $\overleftarrow{X}$ denotes the reversed ordering of $\overrightarrow{X}$.

Let $[\ell]$ represent the set of positive natural numbers up to ℓ for any positive integer ℓ. A voting rule r is called *anonymous* if, for every preference profile $(\succ_i)_{i\in[n]} \in \mathcal{L}(\mathcal{C})^n$

and permutation σ of $[n]$, we have $r((\succ_i)_{i\in[n]}) = r((\succ_{\sigma(i)})_{i\in[n]})$. A voting rule is called *efficient* if the winner can be computed in polynomial time of the input length.

A scoring rule is induced by an m-dimensional vector, $(\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}^m$ with $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_m$ and $\alpha_1 > \alpha_m$. A candidate gets a score of α_i from a voter if she is placed at the i^{th} position in the voter's preference order. The score of a candidate from a voter set is the sum of the scores she receives from each of the voters. If α_i is 1 for $i \in [k]$ and 0 otherwise, we get the k-approval rule. If α_i is 0 for $i \in [m-k]$ and -1 otherwise, we get the k-veto rule. The scoring rules for score vectors $(1, 0, \ldots, 0)$ and $(0, \ldots, 0, -1)$ are called plurality and veto rules respectively. The scoring rule for score vector $(m-1, m-2, \ldots, 1, 0)$ is known as the Borda rule.

The scores for the other voting rules studied in the paper (apart from scoring rules) are defined as follows. Let $vs(a, b)$ be the difference in the number of votes in which a precedes b and the number of votes in which a succeeds b, for $a, b \in \mathcal{C}$. The maximin score of a candidate a is $\min_{b\neq a} vs(a, b)$. The candidate with the maximum maximin score (after tie-breaking) is the winner. Given $\alpha \in [0, 1]$, the Copeland$^\alpha$ score of a candidate a is $|\{b \neq a : vs(a, b) > 0\}| + \alpha \times |\{b \neq a : vs(a, b) = 0\}|$. The candidate with the maximum Copeland$^\alpha$ score (after tie-breaking) is the winner. We will assume α to be zero, if not mentioned separately. A candidate a that has a positive pairwise score $vs(a, b)$ against all other candidates $b \in \mathcal{C}$ is called a Condorcet winner. Rules which select a Condorcet winner as the winner (whenever it exists) are called Condorcet-consistent rules. Copeland and maximin voting rules are Condorcet-consistent. The simplified Bucklin score of a given candidate a is the minimum number k such that more than half of the voters rank a in their top k positions. The candidate with the lowest simplified Bucklin score (after tie-breaking) is the winner and her score is called the Bucklin winning round.

We use $s(c)$ to denote the total score that a candidate $c \in \mathcal{C}$ gets in an election. Similarly, if $Y \subseteq \mathcal{C}$, we use $s(Y)$ to refer to the score of each candidate from Y; e.g. $s(Y) = 5$ means that each candidate from Y has a score of 5. The voting rule under consideration will be clear from the context. Note that we assume that the briber is aware of the votes cast by each voter.

2.3 Our problem definition

We now define our problems formally. Let r be a voting rule.

Definition 2.1 (Effect of Applying Shift Vector on a Network of Voters)**.** Suppose we have a set $\mathcal{C}$ of m candidates, a set $\mathcal{V}$ of n voters, an n-voter profile $\mathcal{P} = (\succ_i^{\mathcal{P}})_{i\in[n]} \in \mathcal{L}(\mathcal{C})^n$ over $\mathcal{C}$, a directed weighted network $\mathcal{G} = (\mathcal{V}, \mathcal{A}, w : \mathcal{A} \longrightarrow \mathbb{R}^+)$, and a shift vector $\mathfrak{s} = (s_1, \ldots, s_n) \in \mathbb{N}_0^n$. We define a tuple $(s'_1, \ldots, s'_n) \in \mathbb{N}_0^n$ as follows.

$$s'_i = s_i + \sum_{j\in[n]\setminus\{i\}:(j,i)\in\mathcal{A}} s_j \cdot w(j,i)$$

Let $c \in \mathcal{C}$ be the candidate we want to make win. Let $\succ_i^{\mathcal{Q}}$ be the preference obtained from $\succ_i^{\mathcal{P}}$ by shifting c to left by $\min\{s'_i, \text{pos}(c)\}$ positions. We call the profile $\mathcal{Q} = (\succ_i^{\mathcal{Q}})_{i\in[n]}$ to be the profile resulting from applying $\mathfrak{s}$ on $\mathcal{G}$.

Problem Definition 1: [Shift Bribery over Network]
Given a set $\mathcal{C}$ of m candidates, a set $\mathcal{V}$ of n voters, an n-voter profile $\mathcal{P} = (\succ_i^{\mathcal{P}})_{i\in[n]} \in \mathcal{L}(\mathcal{C})^n$ over $\mathcal{C}$, a directed weighted network $\mathcal{G} = (\mathcal{V}, \mathcal{A}, w : \mathcal{A} \longrightarrow \mathbb{R}^+)$, the preferred candidate $c \in \mathcal{C}$, a preference $\succ_B \in \mathcal{L}(\mathcal{C})$ of the briber, a family $\Pi = (\pi_i : [m-1] \to \mathbb{N})_{i\in[n]}$ of cost functions (where $\pi_i(0) = 0, \forall i \in [n]$) corresponding to every voter, and a budget $b \in \mathbb{R}$, compute if there exists a shift vector $\mathfrak{s} = (s_1, \ldots, s_n) \in \mathbb{N}_0^n$ such that: (1) $\Sigma_{v_i\in\mathcal{V}}\pi_i(s_i) \leqslant b$ (2) the candidate c is a winner in the profile which is obtained by applying the shift vector $\mathfrak{s}$ on $\mathcal{G}$. An arbitrary instance of Shift Bribery over Network is denoted by $(\mathcal{C}, \mathcal{P}, \mathcal{G}, c, \succ_B, \Pi, b)$.

Problem Definition 1 is the constructive version of the problem. We can also as well define and study the destructive version of the problem.

Chapter 3

Results

3.1 Complete Graph with all edge weights 1

Theorem 3.1. Shift Bribery over Network *is polynomial-time solvable if the underlying graph is complete and all the edge weights are 1 for the majority voting rule, and linear cost functions.*

Proof: Let $(\mathcal{C}, \mathcal{P}, \mathcal{G}, c, \succ_B, \Pi, b)$ be an arbitrary instance of our problem, under the majority voting rule. Let us say the majority rule elects a dummy candidate in the event no majority is achieved. The goal of bribery is to then make our candidate c a Condorcet winner. The approach towards the solution is greedily solved. Notice that for all shift vectors $\mathfrak{s} = (s_i)_{i\in[n]}$, the value of $s'_i \ \forall i \in [n]$ is same and equal to $\sum_{i\in[n]} s_i$. Let us call this quantity $\mathcal{S}$.

Let us say the position vector of candidate c is $\mathfrak{p} = \left(\succ_i^{\mathcal{P}}(c)\right)_{i\in[n]}$ where $\succ_i^{\mathcal{P}}(c)$ specifies the position of desired candidate c in the ranking $\succ_i$ in preference profile $\mathcal{P}$. Notice that c is a condorcet winner $\iff \mathcal{S} \geqslant$ the median of this position vector. This is because all voters will see a shift of $\mathcal{S}$, and only those voters with $\succ_i^{\mathcal{P}}(c) \leqslant \mathcal{S}$ will vote for c under the plurality scoring rule.

Hence the minimum value of $\mathcal{S}$ needed to make c a condorcet winner can be easily found using any polynomial algorithm to find median in a vector. Let us say the

family of linear cost functions for each voter are $\Pi = (\pi_i : [m-1] \to \mathbb{N})_{i \in [n]}$ (where $\pi_i(x) = b_i \times x, \forall i \in [n]$), and a budget $b \in \mathbb{R}$. This value of $\mathcal{S}$ can be achieved with minimum possible budget by bribing only the voter with the smallest cost coefficient, at a cost of $b_i \times \mathcal{S}$, (where b_i is smallest among all $i \in [n]$). This is the minimum possible cost to make c a condorcet winner. If this cost is $\leqslant b$, then c can be made a condorcet winner. If not, c cannot be made a condorcet winner.

□

Let us now see the same problem with the plurality voting rule instead of the majority voting rule

Lemma 3.2. *define $\mathcal{S}$ as above. If c is a winner under the plurality voting rule for some $\mathcal{S}'$, then c is also a winner for all $\mathcal{S} \geqslant \mathcal{S}'$*

Proof: As discussed previously, all voters in this problem framework shift by the same maximum amount $\mathcal{S}$. Let us say c wins for a given value of $\mathcal{S}$. This means that c has more votes than all other candidates $c' \in \mathcal{C}$. We prove that this cannot change upon increasing $\mathcal{S}$. Notice that increasing the value of $\mathcal{S}$ cannot reduce the number of votes c gets, since all voters now shift c to a more preferable location. Similarly, Notice that increasing $\mathcal{S}$ cannot increase the number of votes a rival of c can get, since all other candidates either stay stationary, or shift to a less preferable profile. This completes the proof.

□

Lemma 3.3. *define $\mathcal{S}$ as above. If c is a winner under the plurality voting rule for some $\mathcal{S}'$, then c is also a winner for all $\mathcal{S} \leqslant \mathcal{S}'$*

Proof: Very analogous to the previous proof. Let us say c wins for a given value of $\mathcal{S}$. This means that c has less votes than some other candidate $c' \in \mathcal{C}$. We prove that this cannot change upon decreasing $\mathcal{S}$. Notice that decreasing the value of $\mathcal{S}$ cannot increase the number of votes c gets, since no voter will shift c to a more preferable location. Similarly, Notice that decreasing $\mathcal{S}$ cannot decrease the number

of votes a rival of c can get, since all other candidates either stay stationary, or shift to a more preferable profile.

□

Lemma 3.4. *The optimal value of* $\mathcal{S}$ *has to be in the position vector* $\mathfrak{p} = \left(\succ_i^{\mathcal{P}}(c)\right)_{i\in[n]}$ *or 0.*

Proof: For simplicity, let us add 0 to our position vector Consider a value of $\mathcal{S}$ that is not equal to any $\succ_i^{\mathcal{P}}(c)$ for any $i \in [n]$ (or 0). Let us assume that this value of $\mathcal{S}$ is optimal (the lowest among all $\mathcal{S}$ that make c win). Now, consider the largest value of an element in our position vector that is smaller than $\mathcal{S}$. Let us call this element p. Notice that reducing the value of $\mathcal{S}$ to p cannot cause out candidate c to lose. This is because all voters with $\succ_i^{\mathcal{P}}(c)$ less than p voted for c previously, and will continue to do so. All the other voters didn't vote for c even with the old value of $\mathcal{S}$ since their $\succ_i^{\mathcal{P}}(c)$ was strictly greater than $\mathcal{S}$. This means that $p \geqslant \succ_i^{\mathcal{P}}(c) \iff \mathcal{S} \geqslant \succ_i^{\mathcal{P}}(c)$, which implies that the value p is also a valid $\mathcal{S}$ that can make c win. since $p < \mathcal{S}$, this contradicts the assumption that $\mathcal{S}$ is optimal.

Note that this proof does not work if we do not add 0 to our position vector, since $\mathcal{S} = 0$ could also be optimal (c is already the winner).

□

Theorem 3.5. Shift Bribery over Network *is polynomial-time solvable if the underlying graph is complete and all the edge weights are 1 for the plurality voting rule, and linear cost functions.*

Proof: for a given $\mathcal{S}$, we can simulate the vote and check if c is a winner in polynomial time. From the **lemma 3.2** and **lemma 3.3**, it follows that we can binary search over possible values of $\mathcal{S}$ to find the smallest value of $\mathcal{S}$ that satisfies our agenda. From **lemma 3.4** it follows that we only need to check $\mathcal{S}$ values from our position vector $\mathfrak{p} = \left(\succ_i^{\mathcal{P}}(c)\right)_{i\in[n]} \cup [0]$. Hence, we need to search from a set of polynomial size, and checking each value of the set also takes polynomial time by generating

new preference profiles for each voter. The time complexity to get the optimal value of $\mathcal{S}$ is therefore $O(n\log(n))$.

Once we have found the optimal value of $\mathcal{S}$, finding the optimal bribery scheme follows the same procedure as before.

The value $\mathcal{S}$ can be achieved with minimum possible budget by bribing only the voter with the smallest cost coefficient, at a cost of $b_i \times \mathcal{S}$. If this cost is $\leqslant b$, then c can be made a plurality winner, else not.

□

3.2 path with all weights 1, and identical linear cost functions

without loss of generality, let us assume the i^{th} voter to be connected to the $i+1^{th}$ by and edge of weight 1. consider that the shift vector $(s'_1, \ldots, s'_n) \in \mathbb{N}_0^n$. It now follows from our problem definition that $s'_i = s_i + s_{i+1}$. under plurality, for a voter i to vote for c, $s'_i \geqslant \succ_i^{\mathcal{P}} (c)$

Lemma 3.6. *There exists an optimal solution where* $s_i > \succ_i^{\mathcal{P}} (c) \Rightarrow s_i + s_{i-1} = \succ_{i-1}^{\mathcal{P}} (c)$

Proof: First consider the case where $s_i + s_{i-1}$ is greater than $\succ_{i-1}^{\mathcal{P}} (c)$. Now let us say there exists an optimal solution with $s_i > \succ_i^{\mathcal{P}} (c)$. We can decrease the value of s_i until $s_i = max(\succ_i^{\mathcal{P}} (c), \succ_{i-1}^{\mathcal{P}} (c) - s_{i-1})$ without changing optimality. This is because we do not decrement s_i past the point where either voter changes their vote. We have now generated a new optimal solution that satisfies the above property.
Now consider the case where $s_i + s_{i-1} < \succ_{i-1}^{\mathcal{P}} (c)$. Once again let us say there exists an optimal solution with $s_i > \succ_i^{\mathcal{P}} (c)$. We can decrease the value of s_i to $\succ_i^{\mathcal{P}} (c)$ without changing optimality. Since the i^{th} voter will continue voting for candidate c and the $i-1^{th}$ voter will continue to not vote for c. the value of s_i does not impact any other voters.
This completes the proof

□

Lemma 3.7. *There exists an optimal solution where* $s_{i+1} > 0 \Rightarrow s_i + s_{i+1} = \succ_i^{\mathcal{P}} (c)$

Proof: let us say $s_{i+1} > 0$ and $s_i + s_{i+1} > \succ_i^{\mathcal{P}} (c)$. We can now reduce the value of s_{i+1} to $max(0, \succ_i^{\mathcal{P}} (c) - s_i)$, while increasing s_{i+2} by the same amount. Notice that this does not change optimality since we do not decrement s_{i+1} enough to cause voter i to not vote for c. Also, since we incremented s_{i+2}, voter i+1 will also not change their vote (since $s_i + s_{i+1}$ has not changed). Subsequent voters can only be better off due to increasing s_{i+2}, since that increases the value of their shifts. Notice that this also does not change the optimality of cost, since all voters have the same cost function, and we are just transferring values of s_i
Now let us say $s_i + s_{i+1} < \succ_i^{\mathcal{P}} (c)$. We can reduce s_{i+1} to 0 and increment s_{i+2} by the same amount. This also yields an optimal solution since it neither changes voter i's vote, nor voter i+1. Subsequent voters can only be better off due to increasing s_{i+2}, since that increases the value of their shifts. Once again, this does not affect the optimality of costs, since all voters have identical cost functions.

□

Lemma 3.8. *There exists an optimal solution where* s_0 *is 0.*

Proof: Similar logic to above, transfer s_0 to s_1. This does not affect $\succ_0^{\mathcal{P}} (c)$, and can only positively affect subsequent voters.

□

From the above 3 lemmas, it follows that for every value s_{i+1} we only need to worry about satisfying $\succ_i^{\mathcal{P}} (c)$. for any s_{i+1}, we can either choose to satisfy the i^{th} voter's shift requirement, or disregard it completely and set s_{i+1} to 0. Also since all cost functions are identical and linear, we only need to worry about minimising $\sum_{i \in [n]} s_i$
For the majority voting rule this gives us a dynamic programming solution as follows:

Algorithm: define $dp[val][i][k]$ to be the minimum budget used until voter i (not included) such that there are k more voters left to achieve majority, and the value

of s_i is val. Initialise $dp[p0][1][ceil(n/2)-1]$ to $\succ_0^{\mathcal{P}}(c)$ (to satisfy the first voter with s_1). Initialise all other values of dp to -1. At each step we can either chose to satisfy another voter, or not:

If we choose to satisfy voter i:
$if(val < \succ_i^{\mathcal{P}}(c))$ then
$dp[\succ_i^{\mathcal{P}}(c) - val][i+1][k-1] =$
$max(dp[\succ_i^{\mathcal{P}}(c) - val][i+1][k-1], dp[val][i][k] + \succ_i^{\mathcal{P}}(c) - val)$

$if(val > \succ_i^{\mathcal{P}}(c))$ (voter i already satisfied) then
$dp[0][i+1][k-1] = max(dp[0][i+1][k-1], dp[val][i][k])$

If we choose not to satisfy voter i:
(val must be less than pi)
$dp[0][i+1][k] = max(dp[0][i+1][k], dp[val][i][k])$

Note that from **Lemma 3.6**, the maximum possible value of any val is $max(\succ_i^{\mathcal{P}}(c))i \in [n]$. This quantity cannot be more than n. Hence the number of dp states is at most $n \times n \times ceil(n/2)$. Solving the dp either recursively or iteratively will lead to a polynomial time solution, since states are not revisited.
Note: There might be more space or time optimised solutions to this problem. This algorithm is simply to show that the problem is not NP-Hard

Chapter 4

Future work and conclusions

4.1 The general problem

The complexity class of the problem is still unknown and needs further research. However many different approaches have been tried, and we strongly believe the problem to be NP-complete.

Conjecture: Shift Bribery over Network under the majority voting rule reduces to Shift Bribery over Network under the plurality rule.

Proof Outline: consider an arbitrary instance of the problem with the majority voting rule. Add a dummy candidate c' as the first preference for every voter and create an instance of the problem with the plurality rule. Note here that for c to be a winner, we must convince a **majority** of voters to vote for c. (Since anything less than a majority will lead to c' winning. **issue with the above approach:** We now need to increase the required shift values for all voters by 1 for a mapping of solutions sets to the constructed instances. While this is trivial in standard shift bribery by simply adjusting the cost functions to map from s_i to s_{i+1}, **the same cannot be done** in Shift Bribery over Network.

Conjecture: The problem in its most general sense under the majority voting rule can be proved to be NP-hard using the MAXIMUM FEASIBLE SUBSYSTEM problem (Edoardo Amaldi1 and Leslie E. Trotter, 2003).

Proof Outline: Define **Problem MFS-I** to be a variant of the MAXIMUM FEASIBLE SUBSYSTEM problem with 1 inequality Q necessarily satisfied, along with at least n other inequalities.

This now seems analogous to our given problem with 1 budget constraint that needs to be satisfied, and n other inequalities of which we need to maximise the number of voters satisfied. The issue with this approach for the plurality rule is that it does not consider the change in rival positions upon satsifying the inequalities. However, even proving a reduction for the majority rule (where rivals no longer matter) proved to be non trivial.

4.2 Conclusions

We could not conclusively find the the complexity class of the general problem. However, special cases of the problem proved simpler to solve and had polynomial solutions.

As mentioned previously, there is no prior literature in computational social choice theory that models the inter-relationship between different voters and their influences. Our problem definition attempts to bridge that gap, while providing new insights into the nature and vulnerability of various electoral systems in the social world.

Bibliography

Bartholdi, J., Tovey, C., and Trick, M. (1989). The computational difficulty of manipulating an election. *Soc. Choice Welf.*, 6(3):227–241.

Bartholdi III, J. J., Tovey, C. A., and Trick, M. A. (1992). How hard is it to control an election? *Math Comput Model*, 16(8-9):27–40.

Bredereck, R., Faliszewski, P., Niedermeier, R., and Talmon, N. (2016). Large-scale election campaigns: Combinatorial shift bribery. *J. Artif. Intell. Res.*, 55:603–652.

Edoardo Amaldi1, M. E. P. and Leslie E. Trotter, J. (2003). Some structural and algorithmic properties of the maximum feasible subsystem problem.

F. Brandt, V. Conitzer, U. E. J. L. and Procaccia, A. (2015). *Handbook of computational social choice*, volume 1. freeman New York.