IMPERIAL COLLEGE LONDON

MEng Individual Project

The Computational Complexity of Bribery in a Network-Based Rating System

James Stewart

supervised by

Dr. Paolo Turrini
Dr. Umberto Grandi

June 21, 2017

Abstract

We study a rating system in which a finite set of customers are able to evaluate some service. A customer's evaluation is influenced by those of its trusted peers where the customers are arranged in a network of influence. A malicious service provider, for example, a restaurant owner, may choose to bribe a subset or all of these customers by reinvesting some of their income so as to increase their overall gain. Recent research has examined bribing strategies within this framework under the assumption that all customers express an evaluation and also has determined in which cases this framework is bribery-proof.

We extend this framework to the case where perhaps some customers do not vote but where these customers can be bribed to vote in future. We look at the computational complexity of the restaurant owner determining optimal bribing strategies. Amongst other things, we prove that the non-voter framework is very different from the all-voter framework as computing the optimal bribing strategy in the former is **NP**-hard whereas in the latter it is polynomial-time solvable. We propose and implement approaches to computing bribing strategies on artificial social networks designed to model real-world networks.

Finally, we look at the case where the restaurant owner has no knowledge of the influence network between customers. We show the **NP**-hardness and **W**[**2**]-hardness of problems closely related to bribery. We see that although the framework is not bribery-proof in general, there are significant complexity-theoretic barriers to being able to efficiently compute bribing strategies.

Acknowledgements

I would like to express my sincere thanks towards both Dr. Paolo Turrini and Dr. Umberto Grandi for the help and guidance that they have given me throughout the course of this project.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Introduction

In fields such as artificial intelligence and game theory we are interested in studying how the actions of a number of rational agents or decision makers can influence each other. More recently however, fields such as social choice theory, and particularly computational social choice theory, have emerged. Social choice theory provides us with a theoretical framework for dealing with aggregating the preferences, goals or opinions of multiple individuals to provide a combined decision. A comprehensive introduction to the field is provided by Brandt *et al.* in their *Handbook of Computational Social Choice* [1]. Related lines of research include the formal study of network based voting and mechanism design, lobbying and bribery and reputation based systems [2].

Proposed by P. Turrini and U. Grandi in their paper *A network-based rating system and it's resistance to bribery* [2] is a rating system in which a finite set of individuals are able to evaluate a service. Within this model, two notions of rating are defined which directly influence the utility of some service provider. The paper investigates the effect upon the network when we allow some malicious service provider to bribe a subset or all of these individuals. Further to this, the effects of bribing strategies under various constraints are analysed and it is shown under which of these the proposed system is bribery-proof. The ideas introduced and the techniques used are fundamentally related to the fields of computational social choice [3] and game theory. Although this is the case, a number of interesting problems arise when we begin to consider the computational complexity of certain aspects of the model introduced. As well as this, contemplating how hard it is to solve these problems has a real practical value and provides a useful extension to the ideas presented in [2].

The overall aim of this project is to extend the network based rating system and it's analysis of bribery under the personalised rating system. That is, we aim to introduce new ideas and algorithms as to how we can compute optimal bribing strategies and provide an analysis of the computational complexity of the various problems which result from imposing realistic network restrictions upon the system. One possible application is to picture a restaurant owner who is able to play bribing strategies on the system in order to influence customer opinion and therefore increase the revenue of their own restaurant. The utility a restaurant receives is proportional to its rating. Two notions of rating are defined: the $\mathbb{O}$-rating which aggregates the individual ratings of all customers and the $\mathbb{P}$-rating which takes into account a network of influence between customers.

From the perspective of the restaurant owner, there are a number of pieces of information that it may have which can influence the effectiveness of any attempted bribe and the complexity of any algorithm that might yield an optimal bribing strategy (in the case of the $\mathbb{P}$-rating). In simple terms, this information is: knowledge of individual customers evaluations, knowledge of the network of customers, and knowledge of customers positions in the network. The paper introduces a polynomial-time algorithm for the computation of optimal bribing strategies in the case that the restaurant has full knowledge. It then discusses the cases where the restaurant might have less information, but leaves these cases largely unexplored and certainly does not explore in any detail the computational complexity of the resulting problems.

Proven in the paper, are the restrictions upon the model which render the network bribery-proof or bribable under the proposed new $\mathbb{P}$-rating. An important concluding remark of the paper is that in general, the $\mathbb{P}$-Rating and $\mathbb{O}$-Rating systems are not bribery proof. As a result of this, we would like to study the intrinsic difficulty of actually computing a bribing strategy. We may eventually find that even in the cases where it is possible for us to define optimal bribing strategies, it might not be computationally feasible to do so. Proving such results would be of significant importance to the proposed rating systems. For example, if we can formally prove that under a certain set of restrictions the problem of computing an optimal bribing strategy is intractable, even if such strategies might exist, we can be confident that the system is suitably resistant to bribery. We see that the paper is primarily concerned with the rating system's resistance to bribery and subsequently shows that this is not necessarily the case. A complexity theoretic approach to proving that it is computationally hard to compute bribing strategies could give some amount closure to the initial problem being researched.

The first steps towards doing so, are to provide an appropriate mathematical formulation of the problems that we are interested in. This involves introducing the appropriate complexity and graph theoretic background required to begin to contemplate whether the problem is tractable. Once we have done this and clearly defined the problem, we can begin to take a deeper look at the problem and the different sets of network restrictions that are outlined in [2]. The aim is to be able to identify for all combinations of network restrictions whether the problem is tractable or not. For the tractable cases, we would like to provide an algorithm in order to compute such strategies in polynomial-time; just as the paper does for the most simple case. For those intractable cases, we should formally prove that the problem is intractable and provide a comprehensive analysis of its complexity. In addition to this, it would be beneficial to research into methods of computing reasonable but not optimal bribing strategies. There are a number of possible approaches to this motivated by a range of different areas such as: heuristic and probabilistic methods or genetic algorithms.

In addition to these largely theoretical results and especially in the context of computing heuristic solutions, we would also like to be able to evaluate the performance of such algorithms and the quality of their results. As a result of this, it would be useful to provide an implementation and extension of the model introduced in the paper. We should also consider realistic methods of generating networks on which to generate and execute bribing strategies. Once we have done this, we can perform computational simulations to evaluate the quality of each approach. This will provide us with experimental results to consolidate our theoretical ideas.

These approaches only consider the problems outlined by the paper, however. There are also a number of appropriate modifications that could be made to the rating system. For example, one such investigation could be the definition of the $\mathbb{P}$-rating. The implications on the computational complexity of the resulting problems could be drastically improved if we consider alternative notions of a localised rating system.

A classical complexity theoretic approach can sometimes offer a less refined analysis of the difficulty of certain problems. Particularly in this case, we consider evaluating the complexity of various problems according to their inherent difficulty with respect to different parameters of the problem. The motivation for these ideas comes from a more modern branch of computational complexity theory known as *parameterized complexity* *theory*. Doing so, we would hope be able to derive results that show we can efficiently compute optimal bribing strategies for small values of some parameter, for example, the number of voting customers. This kind of analysis is of practical value as we can seek to parameterize the problem to reflect commonly occurring scenarios.

1.2 Contributions

The contributions of this project consist of a detailed analysis and further extension to the research presented in [2] from a complexity-theoretic perspective that had not previously been considered. The main contributions of our project are as follows.

1. **Extended the framework in the case of non-voters and known locations (NVKL).** In order to compute a weakly-dominant strategy in the case of full knowledge (AVKL), [2] gives us a natural polynomial-time algorithm. A significant amount of work is first required before we attempt to loosely emulate this approach towards the analogous problem under NVKL. (see Section 5.2)

2. **Proven that the problem of computing a weakly-dominant strategy is NP-complete under NVKL**. We formulate NVKL as a decision problem and are able to show it is NP-complete by a reduction from the independent set problem on 3-regular graphs. We firstly prove several properties concerning the manipulation of bribing strategies. (see Section 5.4)

3. **Developed algorithms to compute profitable bribing strategies for the intractable cases.** We discuss and develop heuristic approaches to computing profitable bribing strategies under NVKL, where the computation of exact solutions is computationally intractable. (see Chapter 6)

4. **Implemented the rating system and performed computational experimentation of our algorithms.** We implement the algorithms and perform an evaluation of their computation results. We discuss their quality and the network features that affect their performance. (see Chapter 6)

5. **Proven the NP-completeness and W[2]-hardness of important related problems given non-voters and unknown locations (NVUL).** We turn our attention to the final case of of NVUL and consider the closely related problem of

utility maximising evaluation placements. We show that this is **NP**-complete and, furthermore, that it is **W**[**2**]-hard. (see Chapter 7)

Chapter 2

Background Theory

2.1 Motivation

The complexity theoretic analysis of bribery in voting systems [4] and systems of *judgement aggregation* [10] (for example elections and lobbying), is becoming an increasingly popular field of research. Systems such as election and voting procedures provide us with a framework for aggregating the preferences and opinions of groups of people. Existing research involves proposing different voting systems and studying the complexity of bribery when we allow an agent to manipulate the judgement aggregation procedure. For example, in [9] we see an investigation into how computationally difficult it is to pay certain voters to change their preferences so that a certain candidate can win the election. In addition to this, alternative lines of research include investigation into the parameterized complexity of bribing judgement aggregation systems with respect to many of the natural and realistic parameterizations that arise. Research of this nature is relevant to our project and an example can be seen in [10].

Models for judgement aggregation and, in particular, the model presented by Turrini and Grandi in [2] are graph-theoretic by nature. A natural way to model influence between a finite number of rational agents is by using a graph. Influence amongst these agents can be represented by a relation which comprises the edges of the graph. The study of graphs, known as *graph theory*, provides us with a rich and extensive range of techniques with which we can study how information or opinion diffuses through such structures. These graphs of rational agents can represent social networks, voter networks, and a range of other types of networks. Throughout the course of this project, we will assume knowledge of the fundamentals of graph theory, algorithmic graph theory, and discrete mathematics. The following section contains an introduction to the essential definitions and algorithms in these areas.

2.2 Graph Theory

We take the following basic definitions from [23]. We have selectively chosen a number of relevant definitions in relation to the techniques we will use and proofs we will give in later chapters of this project. Further explanation and related facts and definitions can be found in the same text.

Definition 2.1. *Graph: A graph is a pair $G = (V, E)$ of sets satisfying $E \subseteq V^2$. The elements of V are the vertices or nodes of G, the elements of E are the edges or arcs of G.*

Definition 2.2. *Order: The order of a graph G is the number of vertices of G, written $|G|$; its number of edges is denoted by $||G||$. An infinite graph G is such that $|G| = \infty$; a graph is said to be finite otherwise.*

Definition 2.3. *Incident: A vertex $v \in V$ is incident with an edge e if there exists $v' \in V$ (possibly with $v = v'$) such that $(v, v') \in E$.*

Definition 2.4. *Degree: The degree of $x \in V$ is the number of vertices in V that are incident to x.*

Definition 2.5. *Two vertices $x, y \in V$ are adjacent or neighbours if and only if $(x, y) \in E$.*

Definition 2.6. *Neighbourhood: The neighbourhood of $x \in V$, or $N(x)$, is the set $\{y \in V : (x, y) \in E\}$.*

Definition 2.7. *Second Neighbourhood: The second neighbourhood of $x \in V$, or $N^2(x)$, is the set $\bigcup\{N(y) : y \in N(x)\}$.*

Definition 2.8. *Clique: A clique in a graph $G(V, E)$ is a subset of of the vertices of V such that all distinct pairs of vertices are adjacent.*

Definition 2.9. *Independent: Pairwise non-adjacent vertices or edges are called independent.*

Definition 2.10. *Independent Set: An independent set is a set of pairwise-independent vertices.*

Definition 2.11. *k-regular: A graph $G(V, E)$ is said to be k-regular (for $k \in \mathbb{N}$) if all vertices of V are of degree k.*

Definition 2.12. *Vertex Dominating Set: A dominating set of a graph $G(V, E)$ is a set of vertices $X \subset V$ such that every vertex in $V \setminus X$ is adjacent to at least one vertex of X.*

2.3 A Network-Based Rating System

Now that we have provided some background around the fundamental concepts and definitions of graph theory, we can now begin to present the network-based rating system. A formal framework for the network based rating system and its associated $\mathbb{O}$-rating and $\mathbb{P}$-rating is presented by Turrini and Grandi. The following section contains the essential definitions, taken entirely from [2].

2.3.1 The Framework

The framework consists of an object r called a *restaurant*. This object can be evaluated by a finite set of *customers* $C = \{c_1, ..., c_n\}$. Connecting these customers is an undirected graph called the *customers network*. Formally, this is a relation $E \subseteq C \times C$. For each $c \in C$, the *Neighbourhood* of c is defined to be $N(c) = \{x \in C : (c, x) \in E\}$. Furthermore, $c \in N(c)\ \forall c \in C$. That is to say that every customer is connected to itself.

Customers concurrently submit an *evaluation* of the restaurant. This evaluation is taken from the set $Val \cup \{*\}$; where $Val \subseteq [0, 1]$ and the distinguished element $(*)$ represents the evaluation of a customer with no opinion (the subtle difference between an evaluation of $*$ and 0 will be discussed later). Note that a property of the chosen set Val is that it is closed under the operation $min\{1, x + y\}$ for all $x, y \in Val$ (where $min\{1, *\} = *$ and $x + * = * + x$ for all $x \in Val$). Most known rating methods, for example a discrete rating scale of one to five stars, can be mapped onto the interval $[0, 1]$ and analysed within this framework.

The actual evaluations provided by customers are given by an evaluation function *eval* which is defined for each customer.

Definition 2.13. *An evaluation is a function $eval : C \to Val \cup \{*\}$.*

Some or possibly all of the customers can express an opinion of the restaurant, and the set of *voters* of the network is defined as $V \subseteq C$ where $V = \{c \in C : eval(c) \neq *\}$. The set of voters is always assumed to be non-empty, i.e, there is always one customer that expresses an evaluation. A customer who expresses an evaluation of 0 differs from one who expresses no evaluation $(*)$ with the former being a member of the set V and the latter not.

In this paper, two rating systems are introduced: the $\mathbb{O}$-rating and the $\mathbb{P}$-rating. It is explained that the $\mathbb{O}$-rating is analogous to rating systems employed by platforms such as Tripadvisor®. Every customer can see the reviews and is therefore influenced by every other customer. In the framework presented, that simply means that $E = C \times C$. This method of rating is defined to be the $\mathbb{O}$-rating, which stands for *objective rating*.

Given an evaluation function *eval* for the restaurant, the associated $\mathbb{O}$-rating is defined as follows

Definition 2.14. $\mathbb{O}$-*rating*$(eval) = \underset{c \in V}{avg}(eval(c))$.

Where *avg* is defined to be the average function over $\mathbb{R}$, disregarding $*$.

Rating systems such as this one simply aggregate evaluations into some combined value across all customers. A more refined approach is defined in [2] which gives a *personalised* rating system that takes into account the network of influence between the set of customers. Customer opinions are only influenced by the other customers that they know and trust. That is, their neighbours within the network of influence. This is perhaps more reflective of real-life situations where we would attribute a greater weighting to the opinion of those we know to reputable, as opposed to trusting some anonymous or unknown individual when we make a decision. The $\mathbb{P}$-rating stands for *personalised rating* for an individual customer c and their evaluation *eval* is defined as follows

Definition 2.15. $\mathbb{P}$-*rating*$(c, eval) = \underset{k \in N(c) \cap V}{avg}(eval(k))$.

Finally, in the case that a customer is not a neighbour of any voter, then the $\mathbb{P}$-rating is not defined. Therefore, it is assumed that every customer of the network is connected to at least one voter. It is also stated that when $E = C \times C$, i.e when every customer influences every customer, then for every $c \in C$ $\mathbb{P}$-rating$(c, eval) = \mathbb{O}$-rating$(eval)$. We see this as follows. Let c be any arbitrary customer of C.

$$E = C \times C \implies \mathbb{P}\text{-rating}(c, eval) = \underset{k \in N(c) \cap V}{avg}(eval(k))$$

$$= \underset{k \in C \cap V}{avg}(eval(k)) = \underset{k \in V}{avg}(eval(k)) = \mathbb{O}\text{-rating}(eval)$$

Customer evaluations are interpreted to indicate a customer's propensity to go to a given restaurant. It is assumed, therefore, that the actual utility a restaurant receives is proportional to the rating that it is given by the customers. For the case of the $\mathbb{O}$-rating we have the following.

Definition 2.16. *The initial utility of the restaurant is defined as* $u^0_{\mathbb{O}} = |C|\mathbb{O}\text{-}rating(eval)$.

It therefore follows that the utilities of the restaurant under $\mathbb{O}$-rating and $\mathbb{P}$-rating (where $E = C \times C$) are equal since. (see Definition 2.22)

$$u^0_{\mathbb{P}} = \sum_{c \in C} \mathbb{P}\text{-rating}(c, eval) = \sum_{c \in C} \mathbb{O}\text{-rating}(eval) = |C|\mathbb{O}\text{-rating}(eval) = u^0_{\mathbb{O}}$$

At the initial stage of the game, the restaurant owner receives u^0 and can decide to invest part of it to influence a subset of customers and improve upon the initial game. Such an investment is referred to as a *strategy.*

Definition 2.17. *A strategy is a function* $\sigma : C \to Val$ *such that* $\sum_{c \in C} \sigma(c) \leq u^O$.

The set of all strategies is referred to as Σ and σ^0 is defined to be the strategy that assigns 0 to all customers. A bribing strategy is any strategy different from σ^0. As was previously mentioned, strategies can be played on customers in order to influence their opinion.

Definition 2.18. *The evaluation* $eval^\sigma(c)$ *after execution of a strategy* σ *is* $eval^\sigma(c) = min\{1, eval(c) + \sigma(c)\}$, *where* $* + \sigma(c) = \sigma(c)$, *if* $\sigma(c) \neq 0$, *and* $* + \sigma(c) = *$, *if* $\sigma(c) = 0$.

Strategies are said to be *efficient* if $\sigma(c) + eval(c) \leq 1$ for all $c \in C$. Further, $B(\sigma) = \{c \in C : \sigma(c) \neq 0\}$ is defined to be the set of bribed customers and V^σ is the set of voters after the execution of a strategy σ.

Definition 2.19. *The change in utility due to the execution of a strategy* σ *is defined as* $u^\sigma_{\mathbb{O}} = |C|\mathbb{O}\text{-}rating(eval^\sigma) - \sum_{c \in C} \sigma(c)$.

A definition of a weakly-dominant (or optimal) strategy is also given between strategies in Σ.

Definition 2.20. *Let* σ *be a strategy. The revenue of* σ *is defined as* $r_{\mathbb{O}}(\sigma) = u^\sigma_{\mathbb{O}} - u^\sigma_0$. σ *is profitable if* $r_{\mathbb{O}}(\sigma) > 0$.

Definition 2.21. *A strategy is weakly-dominant if $u^{\sigma}_{\mathbb{O}} \geq u^{\sigma'}_{\mathbb{O}}$ for all $\sigma' \in \Sigma$. It is strictly dominant if $u^{\sigma}_{\mathbb{O}} > u^{\sigma'}_{\mathbb{O}}$ for all $\sigma \in \Sigma$.*

All of these given definitions are easily adapted to the case of the $\mathbb{P}$-rating.

Definition 2.22. $u^{0}_{\mathbb{O}} = \sum\limits_{c \in C} \mathbb{P}\text{-}rating(c, eval)$.

Definition 2.23. $u^{\sigma}_{\mathbb{O}} = \sum\limits_{c \in C} \mathbb{P}\text{-}rating(c, eval^{\sigma}) - \sum\limits_{c \in C} \sigma(c)$.

Definition 2.24. $r_{\mathbb{P}}(\sigma) = u^{\sigma}_{\mathbb{P}} - u^{\sigma}_{0}$.

2.3.2 Bribing the System

Having formally defined this system, the we may now discuss bribing strategies under both the $\mathbb{P}$-rating and $\mathbb{O}$-rating. A polynomial-time algorithm is given in [2] for the case that the restaurant knows eval, the network and the positions of the customers on the network. In high-level terms, this algorithm is a greedy algorithm which selects customers to bribe at each stage of the algorithm depending on their so-called *influence weight*. It is defined as follows

Definition 2.25. *The influence weight of a customer $c \in C$ in a network E and a set of designated voters V is defined as follows*

$$w^{V}_{c} = \sum_{k \in N(c)} \frac{1}{|N(k) \cap V|}.$$

Since one assumption of the framework is that every customer can see a voter then this is well-defined. The paper provides, in Lemma 6 [2], an expression for the utility gained by playing a strategy in terms of the influence weights of the customers.

2.4 Computational Complexity Theory

Complexity Theory is the area of computer science that that contemplates the reasons why some problems are so hard to solve by computers [5]. It is one of the most vibrant and active areas of research in modern day theoretical computer science. Complexity theory serves to define and study the intrinsic properties of classes of problems which have similar complexity with respect to a specific model of computation (for example, a Turing machine) and a complexity measure [6].

Complexity theory has a wide range of practical applications. Using it and its many interesting results, we are able to define whole classes of problems which are, for example, tractable, intractable, efficiently parallelizable, or even tractable with respect to some bounded probabilistic error [7]. This has many useful applications to areas such as cryptography, quantum computation, and even biology. Certainly in the context of the problems we are considering, that is, the computation of optimal bribing strategies it is essential to understand the feasibility of any approach to computing one.

From our perspective, we don't necessarily have to explore some of the more structural properties of complexity theory. We would merely be interested in whether or not a problem is tractable or not. In order to understand what it means for a problem to be tractable, this requires a reasonable amount of background from complexity theory.

2.4.1 Big-$\mathcal{O}$ Notation

Before we proceed, we first have to define the big-$\mathcal{O}$ notation. This notation encapsulates the fact that two functions have the same rate of growth. The definition comes from Papadimitriou's *Computational Complexity* [5].

Definition 2.26. *Let f and g be functions from $\mathbb{N}$ to $\mathbb{N}$. We write $f(n) = \mathcal{O}(g(n))$ if there are positive integers c and n_0 such that, for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$.*

2.4.2 Complexity Classes

One aspect of complexity theory is classify problems into complexity classes. These are classes of problems of related complexity with respect to some resource. Most commonly, this resource is space (the amount of memory required to solve a problem) or time (the amount of time required to solve a problem). For the most part of our analysis, we are only concerned with time complexity. Therefore, a more stringent definition of a complexity class is as follows.

The set of all problems which can be solved by an algorithm in $\mathcal{O}(f(n))$ time on an input of size n.

We have used the term '*solve*' rather loosely thus far. In order to precisely define what we mean here, and therefore precisely define exactly what a complexity class is, we must first introduce the concept of a *decision problem.* This is a definition that is extremely important in the area of complexity theory. In fact, the complexity classes that we will define are classes of decision problems. A decision problem is a problem that admits *yes* or *no* as an answer. Formally, we have the following definition which is taken from Bovet and Crescenzi's *Introduction to the Theory of Complexity* [6].

Definition 2.27. *A decision problem* Π *is a triple* $\langle I, S, \pi\rangle$ *such that*

- I *is a set of words that encode instances of the problem*
- S *is a function that maps an instance* $x \in I$ *into a finite set* $S(x)$ *of words that encode possible solutions of* x
- π *is a predicate such that, for any instance* x *and for any possible solution* $y \in S(x), \pi(x, y) = \texttt{true}$ *if and only if* y *is a feasible solution.*

Solving a decision problem $\langle I, S, \pi\rangle$ consists of deciding, for a given instance $x \in I$, whether the set $\{y : y \in S(x) \wedge \pi(x, y)\}$ is not empty.

Finally, before we define the fundamental time complexity classes, we must give one more set of definitions taken from [5].

Definition 2.28. ***TIME***(f) *is the class of all decision problems solvable by a deterministic algorithm in time* $O(f(n))$ *where* n *is the size of the input.*

Definition 2.29. ***NTIME***(f) *is the class of all decision problems solvable by a non-deterministic algorithm in time* $O(f(n))$ *where* n *is the size of the input.*

Now we can define the two most fundamental complexity classes **P** and **NP**.

Definition 2.30. $\boldsymbol{P} = \textbf{\textit{TIME}}(N^k) = \bigcup_{j>0} \textbf{\textit{TIME}}(n^j)$.

Definition 2.31. $\boldsymbol{NP} = \textbf{\textit{NTIME}}(N^k) = \bigcup_{j>0} \textbf{\textit{NTIME}}(n^j)$.

Due to various properties of the class **P**, it is considered to be the best candidate for the class of problems that are feasibly solvable.

Definition 2.32. *A problem is said to be tractable if and only if it is in* ***P****.*

2.4.3 Problem Reduction

An inherent property of complexity classes is that the problems they contain are all of related time complexity. Exactly what we mean by related is that for some complexity class $\mathcal{C}$ and for any two problems A and B in $\mathcal{C}$ then A is *at least as hard as* B. As discussed by Papadimitriu [5] a precise notion of what it means for a problem to be at least as hard as another is *reduction*. A problem A is at least as hard as a problem B if B reduces to A.

Definition 2.33. *B reduces to A if there exists a transformation R which, for every input x of B, produces an equivalent input $R(x)$ of A.*

Here equivalence is described to mean that the answer to $R(x)$ considered as an input for A, 'yes' or 'no', is a correct answer to x, considered as an input of B.

Definition 2.34. *An efficient reduction is one which uses a transformation that can be computed in deterministic polynomial-time.*

Before we begin the next section, the following theorem is essential. We take the theorem from [5] where its proof is also given.

Theorem 2.1. *If R is a reduction from problem P_1 to P_2 and R' is a reduction from problem P_2 to P_3 then $R \cdot R'$ is a reduction from problem P_1 to P_3.*

Given the previous theorem, this gives us an ordering on problems with respect to their difficulty.

2.4.4 Hardness and Completeness

Definition 2.35. *Let $\mathcal{C}$ be a complexity class and let P be a problem. We say that P is $\mathcal{C}$-hard if any problem $P' \in \mathcal{C}$ can be efficiently reduced to P.*

Definition 2.36. *Let $\mathcal{C}$ be a complexity class and let P be a problem in $\mathcal{C}$. We say that P is $\mathcal{C}$-complete if any problem $P' \in \mathcal{C}$ can be efficiently reduced to P.*

Combined with our earlier definitions of **P** and **NP** These two definitions directly yield the complexity classes **NP**-hard and **NP**-complete.

Intuitively speaking **NP**-hard contains the problems that are at least as hard as the hardest problems in **NP**. **NP**-complete contains the problems that can be verified in polynomial-time but for which there is no known way to compute a solution in polynomial-time.

Definition 2.37. *An problem is intractable if it is **NP**-hard.*

Note that if a problem is **NP**-complete then trivially it is **NP**-hard as well.

2.5 Parameterized Complexity Theory

In classical complexity theory, we analyse the complexity of problems in terms of the amount of time or space that is required in order to compute a solution. In some cases, this analysis is not refined enough and it can be misleading in a sense that we could make certain computational problems out to be harder than they actually are. The field of parameterized complexity theory focuses on analysing problems with respect to their difficulty in terms of both the input size and some *parameter* (some function of the input instance).

Operating under the assumption that $\mathbf{P} \neq \mathbf{NP}$, then there exist computational problems for which there is no polynomial-time (in terms of the input size) algorithm to solve. Although this might be the case, that is not to say that there do not exist algorithms which are polynomial-time in the input size and super-polynomial in terms of some parameter κ. If we know that κ will always remain relatively small, such problems are deemed to be tractable in parameterized complexity theory. In particular, we say that such problems are *fixed-parameter tractable*. An example of the fixed-parameter tractability of an **NP**-hard problem can be found in Example 2.1.

Example 2.1. *Consider the satisfiability problem for propositional logic. We parameterize the problem by the number of variables of the input formula. Let κ denote the number of variables of the formula, and let n denote the size of the input (number of clauses). The obvious brute force approach of checking every possible combination of assignments to the variables of the formula has a time complexity of $\mathcal{O}(2^k \cdot n)$. We therefore say that SAT is fixed-parameter tractable.*

The following introduction, and set of essential definitions, will provide a more formal overview of the field. We take the following definitions from [22]. Firstly, we formally define the notion of *parameterization.*

Definition 2.38. *Let Σ be a finite alphabet.*

1. *A parameterization of Σ^* is a mapping $\kappa : \Sigma^* \to \mathbb{N}$ that is polynomial-time computable.*
2. *A parameterized problem (over Σ) is a pair (Q, κ) consisting of a set $Q \subseteq \Sigma^*$ of strings over Σ and a parameterization κ of Σ^*.*

From here we proceed to define the notion of fixed-parameter tractability.

Definition 2.39. *Let Σ be a finite alphabet and $\kappa : \Sigma^* \to \mathbb{N}$ a parameterization.*

1. *An algorithm $\mathbb{A}$ with input alphabet Σ is an fpt-algorithm with respect to κ if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial $p \in \mathbb{N}_0[X]$ such that for every $x \in \Sigma^*$, the running time of $\mathbb{A}$ on input x is at most*
$$f(\kappa(x)) \cdot p(|x|).$$
2. *A parameterized problem (Q, κ) is fixed-parameter tractable if there is an fpt-algorithm with respect to κ that decides Q. FPT defines the class of all fixed-parameter tractable problems.*

Just like in classical complexity theory, we may also define a notion of reduction between parametereized problems as follows.

Definition 2.40. *Let (Q, κ) and (Q', κ) be parameterized problems over the alphabets Σ and Σ', respectively. An fpt-reduction from (Q, κ) to (Q', κ) is a mapping $R : (\Sigma)^* \to (\Sigma')^*$ such that:*

1. *For all $x \in \Sigma^*$ we have $(x \in Q \iff R(x) \in Q')$.*
2. *R is computable by an fpt-algorithm (with respect to κ). That is, there is a computable function f and a polynomial $p(X)$ such that $R(x)$ is computable in time $f(\kappa(x)) \cdot p(|x|)$.*

3. *There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $\kappa'(R(x)) \leq g(\kappa(x))$ for all $x \in \Sigma^*$.*

The class FPT is closed under fpt-reductions [22]. Whilst FPT is the class of fixed-parameter tractable problems. We can define other important classes of problems in parameterized complexity theory as follows.

Definition 2.41. 1. *Let Σ be an alphabet and $\kappa : \Sigma^* \to \mathbb{N}$ a parameterization. A nondeterministic Turing machine $\mathbb{M}$ with input alphabet Σ is called κ-restricted if there are computable functions $f, h : \mathbb{N} \to \mathbb{N}$ and a polynomial $p \in \mathbb{N}_0[X]$ such that on every run with input $x \in \Sigma^*$ the machine $\mathbb{M}$ performs at most $f(\kappa) \cdot p(n)$ steps, at most $h(\kappa) \cdot log(n)$ of them being nondeterministic. Here $n = |x|$ and $k = \kappa(x)$.*

2. ***W[P]** is the class of all parameterized problems (Q, κ) that can be decided by a κ-restricted nondeterministic Turing machine.*

Finally, we note that FPT is contained in (and possibly equal to) **W[P]** and **W[P]** is closed under fpt-reductions [22].

2.6 Algorithms

The computation of weakly-dominant bribing strategies presented in the paper uses several fundamental concepts from the theory of algorithms. These concepts include time and space complexity and big-$\mathcal{O}$ notation, which we have already seen. We have seen from Algorithm 1 of [2] that a *greedy algorithm* yields a provably optimal bribing strategy. We will define the concept of a greedy algorithm and provide some background as it is central to our analysis of the existing algorithms and an important factor to consider in future algorithms that we will develop.

In addition to this, our analysis seeks to investigate the tractability of certain problems that arise from the network based rating system. It could be the case that we find certain problems to be intractable, but that should not necessarily hinder any attempts to solve the problem in a manner which provides us with *good* but not necessarily optimal bribing strategies. One way to obtain these is through the use of heuristics.

2.6.1 Greedy Algorithms

When we compute a weakly-dominant bribing strategy we seek to optimise the amount of revenue obtained by playing some strategy on upon a network of customers. Particularly in the case of Algorithm 1 of [2], this involves a sequence of steps at each of which we make a choice with regards to how much we will bribe each customer. Algorithm 1 is a greedy algorithm for the reason that it bribes each customer with the maximal amount possible (so as not to waste any resources) at each step. That is, in terms of how much to bribe each customer at each step, it makes the locally optimal choice. It just so turns out that repeated application of these locally optimal choices leads to a globally optimal solution. This approach leads to the weakly-dominant strategy and so it is an important approach to consider as we attempt to develop algorithms to solve the same problem in each of the various different cases.

2.6.2 Heuristic Methods

Even if we can formally prove that the problem of computing a weakly-dominant bribing strategy under some set of network restrictions is computationally hard, all hope is not lost in terms of finding a *reasonable* solution to the problem instead. Exactly what is meant by reasonable is yet to be defined and it is dependent on the context. For example, a reasonable solution might be a profitable bribing strategy that is not necessarily optimal. Alternatively, it might be a bribing strategy that is provably within some threshold of the truly optimal one.

Use of heuristics enables us to achieve such solutions. A heuristic can be a guess or something which can restrict the search space when computing approximate solutions to a problem. One example of a heuristic, in certain cases, would be a greedy algorithm. Greedy algorithms don't always yield optimal solutions, often they give us reasonable but non-exact solutions. In these cases we could say that it acts as a kind of heuristic.

2.6.3 Polynomial-Time Bribing Strategies

One of the main aims of the research in [2] is stated to be:

"We analyse the effect of bribing strategies under various constraints, and we show under what conditions the system is bribery-proof, i.e., no bribing strategy yields a strictly positive

expected gain to the service provider".

In order properly answer these questions, we must also investigate the feasibility of computing bribing strategies. As we have already discussed in the previous section, feasibly computing a bribing strategy is to compute in polynomial-time (an algorithm in **P**) a bribing strategy. A main aim of this project is to determine whether we can find polynomial-time algorithms to compute weakly-dominant bribing strategies for the various problems that arise when additional restrictions are placed upon the network.

Chapter 3

Lines of Research

3.1 Theoretical Results

There are various interesting analytic and algorithmic problems resulting from the research in [2], some of which we outline below.

3.1.1 All Vote Known Locations (AVKL)

AVKL scenario: The restaurant knows the customer network $E \subseteq C \times C$ as well as the location of each customer in the network and their individual evaluations via $eval : C \to Val$ (note that the range of $eval$ is Val as opposed to $Val \cup \{*\}$ as all customers vote); consequently, we identify the names of customers with the names of network vertices.

AVKL synopsis of [2, Sect. 4.1]: With the definition of $\mathbb{P}$-rating given above, in [2] it is proven that the revenue accrued from any efficient strategy is

$$\sum_{c \in C} (w_c - 1)\sigma(c),$$

and consequently a polynomial-time (greedy) algorithm, called $\mathbb{P}$-greedy, is devised that yields a weakly-dominant bribing strategy. In particular, this shows that computing a weakly-dominant bribing strategy can be done in polynomial-time.

In spite of the conclusive outcome from [2], there are still some related problems to be considered.

The general AVKL scenario can be made more complex. At present, the definition of $\mathbb{P}$-rating takes an average of the evaluations across all neighbours of some vertex (every vertex is always a neighbour of itself, remember). However, there are alternative natural aggregations of evaluations; for example, the maximum, the minimum, the sum of the two largest values, or a weighted average of some sort. Such definitions are motivated by the different ways in which customers pay attention to the evaluations of their neighbours within the network. Potentially, any *symmetric* function might be employed (that is, one where the order of the inputs is of no significance, as is the case for the four suggestions above) as this fits within our graph-theoretic framework.

Problem 1: *For which symmetric functions can we obtain a polynomial-time algorithm to compute a weakly-dominant bribing strategy*?

Alternatively, we can extend the 'sphere of influence' of the $\mathbb{P}$-rating. At present, $\mathbb{P}$-rating$(c, eval)$ is determined by the neighbourhood $N(c)$ of some customer c. We might extend the sphere of influence to include all neighbours of neighbours of c; so, we might define an alternative $\mathbb{P}$-rating as

$$\mathbb{P}'(c, eval) = \frac{1}{|N^2(c) \cap V|} \sum_{k \in N^2(c) \cap V} eval(k),$$

where $N^r(c)$ consists of those vertices at a distance at most r from c. We might even weight the influences of the different vertices of $N^2(c)$ according, perhaps, to their distance from c. Such a definition is motivated by the reasonable scenario where any customer c would take: his or her evaluation as dominating over those of his or her neighbours; and the evaluations of his or her neighbours over their neighbours (that aren't neighbours of c).

Problem 2: *For which $\mathbb{P}$-ratings and 'spheres of influence' can we obtain a polynomial-time algorithm to compute a weakly-dominant bribing strategy*?

There is another natural extension to the basic AVKL scenario. Suppose that the network is a *weighted* network where the weights on the edges denote the strengths of the influence between the two corresponding customers. This relationship strength might feature in the definition of the $\mathbb{P}$-rating. For example, suppose that edge-weights are values from $[0, 1]$.

An amended $\mathbb{P}$-rating might be defined as

$$\mathbb{P}'\text{-rating}(c, eval) = \frac{1}{|N(c) \cap V|} \sum_{k \in N(c) \cap V} \psi_{c,k} \cdot eval(k),$$

where the weight of the edge (c, k) is $\psi_{c,k}$. We might even think of the network graph as being directed so that the influence of customer c over customer c' is potentially different from the influence of customer c' over customer c. This makes sense where two customers are influenced by one another with one perhaps deemed to be an expert, but the other an occasional restaurant visitor with limited food knowledge.

Problem 3: *Can we obtain a polynomial-time algorithm to compute a weakly-dominant bribing strategy when we have weights on the edges of the network?*

There are numerous variations and amalgamations of the three problems defined above.

3.1.2 All Vote Unknown Locations (AVUL)

AVUL scenario: The restaurant knows the customer network $E \subseteq C \times C$ and an evaluation function $eval : N \rightarrow Val$, where N is the set of customer names with $|N| = |C|$. However, the restaurant does not know the locations of the customers in the network; that is, there is no bijection of N to C given. Consequently, if the restaurant were to bribe a customer then the restaurant would have no idea of how this bribe influences the $\mathbb{P}$-ratings of the customer or the neighbours of the customer.

AVUL synopsis of [2, Sect. 4.2]: It was proven in [2, Sect. 4.2] that in expectation, the revenue obtained by any bribing strategy is 0. Hence, no bribing strategy is profitable in expectation.

Note the different nature of the results in the AVUL case. The restaurant does not know the locations of the customers and, in general, there is an exponential number of possibilities to be considered. The restaurant cannot hope to find a weakly-dominant bribing strategy as was the case in the AVKL scenario as it does not know where the customers are located. The evaluation of any bribing strategy has to be aggregated across all possible placements of customers.

The various $\mathbb{P}$-ratings and weight-based extensions from Section 3.1.1 feed into the scenario here.

Problem 4: *What are the expectations of revenue for bribing strategies under the variations proposed in Section 3.1.1?*

The unknown elements of the AVUL scenario open up the possibility of associated problems being computationally hard. The focus in [2, Sect. 4.2] was on averaged revenues. However, it is feasible that a restaurant might wish to have more information on the utilities arising from the possible locations of customers (given their evaluations) in some AVUL scenario. For example, if the maximum utility over all placements of customers was below some threshold that the restaurant deemed low, the restaurant would know that no matter where the customers were actually located, not much utility was coming back to the restaurant and consequently a bribing strategy should be employed to do something about this (under the assumption that there was ample scope to increase the utility).

Problem 5: *What is the maximum utility to be gained across all possible placements of customers?*

Of course, alternative questions might be asked, *e.g.*, 'minimum' might replace 'maximum'. An even more difficult problem to solve is the following one.

Problem 6: *What is the maximum revenue to be gained across all possible placements of customers?*

Note that the maximum revenue to be gained need not come from the placement of customers to locations so as to maximise the utility. Also, customers must be placed before the revenue is calculated as the utility arising from this placement is required so as to limit the total amount of money that can be spent on bribes. Our motivation for posing this problem is that there could be additional costs to a restaurant to actually make the bribes. If the revenue to be gained is not too large then the restaurant could decide that it is not worthwhile to actually make the bribes (we return to such a scenario later when we consider amending the basic model).

Irrespective of whether we have computationally hard problems or not, obvious greedy algorithms might be applied by using, for example, the ranking of the evaluations and the degrees of the vertices in the network.

Problem 7: *How good are basic greedy algorithms in terms of estimating the maximum utility or the maximum revenue to be gained across all possible placements of customers?*

If it turns out that computing the maximum utility or maximum revenue in Problems 5 and 6 is computationally hard then we will need to undertake computational experiments to gain some insight (we discuss these computational experiments later).

Again, irrespective of whether we have computationally hard problems or not, it might be easier to compute the utility or revenue on certain classes of networks. For example, when the network is the complete graph, there is only one placement of customers up to isomorphism. Consequently, we are in the AVKL scenario.

Problem 8: *For which classes of networks is it easy to compute, for example, the average or maximum utility or revenue, or perhaps a good approximation*?

As before, there are numerous variations and amalgamations of the problems defined above.

3.1.3 Non Voters Known Locations (NVKL)

NVKL scenario: The restaurant knows the customer network $E \subseteq C \times C$ and an evaluation function $eval : C \to Val \cup \{*\}$; so, the set of voters $V = \{c \in C : eval(c) \neq *\}$. Again, as the customer locations are known we identify a customer with a network vertex.

NVKL synopsis of [2, Sect. 4.3]: An example was given to show that bribing non-voters might result in additional revenue. The only result proven is that for any efficient bribing strategy where only voters are bribed, the revenue accrued is

$$\sum_{c \in V} (w_c^V - 1)\sigma(c).$$

It is not mentioned as to whether this result can be employed to obtain a weakly-dominant bribing strategy.

Problem 9: *Does a greedy algorithm similar to $\mathbb{P}$-greedy yield a weakly-dominant bribing strategy when we restrict to bribing strategies that bribe voters only*?

The key difficulty in the NVKL scenario is that when a non-voter c is bribed, this bribe can lessen the $\mathbb{P}$-rating of a neighbour of c. At present, there is no analytic understanding of this phenomenon.

Problem 10: *Can we obtain efficient algorithms to compute a weakly-dominant bribing* *strategy? If we cannot obtain exact algorithms, can we obtain algorithms that yield a good approximation?*

As before, there are numerous variations and amalgamations of the problems defined above.

3.1.4 Non Voters Unknown Locations (NVUL)

NVUL scenario: The restaurant knows the customer network $E \subseteq C \times C$ and an evaluation function $eval : N \to Val \cup \{*\}$, where N is the set of customer names with $|N| = |C|$; so, the set of voters $V = \{c \in C : eval(c) \neq *\}$. However, the restaurant does not know the locations of the customers in the network; that is, there is no bijection of N to C given.

NVUL synopsis of [2, Sect. 4.4]: An example is given in [2, Sect. 4.4] showing that $\mathbb{P}$-rating is not bribery-proof in expectation. However, apart from this example, the analysis of the NVUL scenario was left open.

We now have two sources of complication: the unknown locations and the non-voters. Consequently, just like Section 3.1.2, we have the potential for computationally hard problems and we have similar questions to ask.

Problem 11: *Can we efficiently compute the expectation of revenue?*

Problem 12: *What is the maximum utility to be gained across all placements of customers?*

Problem 13: *How good are basic greedy algorithms in terms of estimating the maximum utility to be gained?*

We have versions of the two problems above but for revenue rather than utility.

Problem 14: *For which classes of networks is it easy to compute the average or maximum utility or revenue, or perhaps a good approximation?*

As before, there are numerous variations and amalgamations of the problems defined above.

3.1.5 Changes to the utility

In [2] the actual customer evaluation is interpreted as the propensity of a customer to go to the restaurant, and subsequent utility yields the financial reward for a restaurant. Two notions of utility are considered: for $\mathbb{O}$-rating, the utility is obtained as the sum of the $\mathbb{O}$-ratings of all customers, where we define the $\mathbb{O}$-rating of a customer as its evaluation; and for $\mathbb{P}$-rating, the utility is the sum of the $\mathbb{P}$-ratings of all customers. Both notions of utility involve a summation across the whole set of customers.

However, we can amend the way the utility is used and calculated. Suppose that we are in the AVKL scenario. Suppose also that the utility gained is proportional to the number of people who visit the restaurant and that there is some threshold $t \in [0, 1]$ so that customer c is deemed to almost surely visit the restaurant if $\mathbb{P}$-rating$(c) \geq t$ (maybe t is 1). So, any evaluation results in a certain number of customers whose $\mathbb{P}$-rating is at least t, and this number of customers is the utility. Now the restaurant is interested in making bribes so as to maximise the number of customers having a $\mathbb{P}$-rating of at least t but so that the amount of money needed to make these bribes is affordable by the restaurant. In this scenario, there is no direct relationship between the money a restaurant has at its disposal and the amount of money it invests in bribes. However, a restaurant might have in mind a number of customers, α say, it wishes to have whose $\mathbb{P}$-rating is at least t and it wants to minimise the amount of money it would have to spend in bribes so as to secure α customers whose $\mathbb{P}$-rating is at least t. This gives rise to new problems, the most obvious of which is the following.

Problem 15: *How easy is it for a restaurant to make bribes so as to secure α customers having a $\mathbb{P}$-rating of at least t but so that no more than β is spent in making these bribes*?

We might also amend the model as regards taking into account costs to make the bribes. Imagine the situation where a restaurant makes some decision as regards which customers are to be bribed. There could well be some sort of cost attached to the physical act of bribing a customer. This cost might be financial, perhaps associated with travelling to meet the customer and make the bribe, or in terms of some other resource such as general time and effort. Moreover, this cost might differ from customer to customer. In the first case, the more customers bribed, the more money has to be found from the initial utility; in the second case, although money does not have to be found from the initial utility, there could well be some threshold on the maximum number of bribes that can be made.

Problem 16: *Explore variations of the basic restaurant-customer bribing model and the* *resulting bribing problems arising.*

Chapter 4

All Vote Known Locations (AVKL)

In the case that all customers of the network vote and the restaurant knows the position of all customers on the network, we were able to develop a provably optimal greedy strategy for computing weakly-dominant bribing strategies. The motivation behind this greedy algorithm is developed in [2] and we shall summarise the key details in this section.

4.1 Key Results

A graph-theoretic measure of influence is defined as follows.

Definition 4.1. *The influence weight of a customer $c \in C$ in a network E with the set of voters V is defined as*

$$w_c^V = \sum_{k \in N(c)} \frac{1}{|N(k) \cap V|}.$$

Remark 4.1. *This function is clearly well-defined by the assumption that every customer c is connected to at least one voter (which may be itself). That is, for all $c \in C$ there exists $x \in V$ such that $(c, x) \in E$.*

This measure of influence is important in understanding which customer's $\mathbb{P}$-ratings contribute most to the overall utility of the network. It also separates the topological properties of a customer and its influence away from any form of evaluation. That is, a customers *influence* is entirely dependent on the topology of the network and independent of the evaluations of the customer or any of its neighbours. It also allows us to express the utility of the network concisely as follows [2].

Theorem 4.1. *The utility obtained by playing σ with $\mathbb{P}$-rating is* $u_{\mathbb{P}}^{\sigma} = \sum_{c \in V^{\sigma}} w_c^{\sigma} \times eval^{\sigma}(c) - \sum_{c \in C} \sigma(c)$.

In addition to this, it is shown that we can exactly characterise the revenue obtained by some bribing strategy σ in terms of just the strategy and the influence weights of the bribed customers. This claim takes the form of Proposition 8 of [2], which we include as follows.

Proposition 4.1. *Let $V = C$, let E be a known network, and let σ be an efficient strategy. Then,* $r_{\mathbb{P}}(\sigma) = \sum_{c \in C}(w_c - 1)\sigma(c)$.

Since $\sigma(c) > 0$ for all $c \in C$, we see that $(w_c - 1)\sigma(c) > 0$ is and only if $w_c > 1$ for all $c \in C$. In other words, it is profitable to bribe a customer c is and only if its influence weight w_c is strictly greater than 1.

4.2 A Polynomial-Time Algorithm for Weakly-Dominant Bribing Strategies

We can observe that in order to maximise the revenue of a strategy, we should bribe the customers with the highest influence weights (which are strictly greater than 1) as much as we efficiently can. This observation eventually gives rise to the following greedy algorithm, as presented in [2].

```
Data: Evaluation function $eval$ and network $(C, E)$.
Result: An efficient bribing strategy $\sigma : C \to Val$.
$budget = u^0_{\mathbb{P}}$;
set $\sigma(c) = 0$ for all $c \in C$;
compute $w_c$ for all $c \in C$;
sort $c \in C$ into descending order $c_0, ..., c_m$ based on $w_c$;
for $i = 0, ..., m$ do
    if $budget \neq 0$ then
        if $w_{c_i} > 1$ then
            $\sigma(c_i) = min\{1 - eval(c_i), budget\}$;
            $budget -= \sigma(c_i)$;
        end
    end
end
return $\sigma$
```

Algorithm 1: An algorithm for computing a weakly-dominant strategy under AVKL.

The above algorithm yields a weakly-dominant bribing strategy and clearly does so in polynomial-time. It is stated that the most costly operation involves computing the influence weights w_c for each customer c. The important point to consider here, is that these weights remain static throughout our computation; that is, we can compute them once at the beginning and use them throughout. As we will see in later chapters, it is the static influence weights which allow us to easily compute a weakly-dominant bribing strategy.

Chapter 5

Non-Voters and Known Locations (NVKL)

5.1 Non-Linearity of $\mathbb{P}$-rating

In this section, we consider the situation where the restaurant has complete knowledge of the customer network but where some of the customers do not vote; moreover, the restaurant is considering whether to bribe a subset or all of the customers. Consequently, we suppose that we are given a network (C, E) and an evaluation *eval* together with a subset $V \subset C$ of customers who have voted (so as to yield the evaluation *eval*). Recall that we always assume that any evaluation is such that every customer is adjacent to at least one voter; of course, this applies to the evaluation given to the restaurant before the restaurant considers whether to make any bribes. In [2], the NVKL analysis amounted to noting that one can find networks where bribing a non-voter is profitable and proving a result that when we only bribe voters, the resulting revenue acquired is a function solely of the network structure, the bribes made, and the distribution of voters (but not the initial evaluation). It was stated in [2] that the 'non-linearity' of the NVKL scenario presents issues.

As soon as we introduce the existence of non-voters, we open up the possibility that the structure of the network can change between bribes (by structure we refer to the voter status of customers as this directly affects the $\mathbb{P}$-rating). If we have some non-voter x, it clearly does not contribute to the influence weight of some customer $y \in N^2(x)$ since $x \notin V$ (recall, we have that $N^2(x) = \bigcup\{N(y) : y \in N(x)\}$). If we then bribe x some amount $b > 0$, by definition after the bribe is made the evaluation of x becomes $* + b = b$; that is $x \in V$ now. Clearly the influence weights w_c^V for $(c \in C)$ of the customers is no longer static, and the extent to which a customer is profitable to bribe changes between bribes.

Previously we knew that this was not the case. No matter which customers were bribed, the influence weight remained static and so it gave us a convenient measure upon which to base a greedy algorithm. It was not necessary, when playing some strategy $\sigma : C_1 \subseteq C \to Val$ to define the order (over $c \in C_1$) in which we should execute σ. Clearly in the case of non-voters, we need to think about this as this is not something that is accounted for by the framework, as it is presented in [2].

5.2 Order of Bribes Under NVKL

Under the existence of non-voters, we have two types of customers: voters and non-voters. We investigate, in these two different cases, how much revenue we will gain if we make a bribe of some positive amount. This helps us to decide the order in which any strategy defined over both voters and non-voters should be played.

We begin by considering the revenue gained by (efficiently) bribing a solitary non-voting customer $x \in C \setminus V$ with a bribe $\sigma(x) = b$.

Proposition 5.1. *Let $V \subseteq C$ give rise to an evaluation eval, let $x \in C \setminus V$, and let σ be an efficient bribing strategy such that $\sigma(x) = b > 0$ and $\sigma(y) = 0$, for all $y \in C \setminus \{x\}$. For each $y \in N(x)$, set $\nu_y = |N(y) \cap V|$. The revenue gained is*

$$b \sum_{y \in N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) - \sum_{y \in N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right).$$

Proof. By definition, we have that

$$r_{\mathbb{P}}(\sigma) = u_{\mathbb{P}}^{\sigma} - u_{\mathbb{P}}^{0} = \left(\sum_{c \in C} \mathbb{P}\text{-rating}(c, eval^{\sigma}) \right) - b - \left(\sum_{c \in C} \mathbb{P}\text{-rating}(c, eval) \right).$$

Given that σ bribes exactly one customer x, only the $\mathbb{P}$-rating of customers of $N(x)$

changes; consequently, we have that

$$r_{\mathbb{P}}(\sigma) = \sum_{y \in N(x)} \Big(\mathbb{P}\text{-rating}(y, eval^{\sigma}) - \mathbb{P}\text{-rating}(y, eval) \Big) - b.$$

By the definition of $\mathbb{P}$-rating, we have that $r_{\mathbb{P}}(\sigma)$ is equal to

$$\sum_{y \in N(x)} \left(\frac{1}{|N(y) \cap V^{\sigma}|} \Big(\sum_{k \in N(y) \cap V^{\sigma}} eval^{\sigma}(k) \Big) - \frac{1}{|N(y) \cap V|} \Big(\sum_{k \in N(y) \cap V} eval(k) \Big) \right) - b.$$

For any $y \in N(x)$, let us denote $|N(y) \cap V^{\sigma}|$ by ν_y^{σ}; so, $\nu_y^{\sigma} = \nu_y + 1$. Hence, we have

$$\begin{aligned}
r_{\mathbb{P}}(\sigma) &= \sum_{y \in N(x)} \left(\left[\frac{1}{\nu_y^{\sigma}} \sum_{k \in N(y) \cap V^{\sigma}} eval^{\sigma}(k) \right] - \left[\frac{1}{\nu_y} \sum_{k \in N(y) \cap V} eval(k) \right] \right) - b \\
&= \sum_{y \in N(x)} \left(\left[\frac{1}{\nu_y^{\sigma}} \sum_{k \in N(y) \cap V} eval(k) \right] + \frac{b}{\nu_y^{\sigma}} - \left[\frac{1}{\nu_y} \sum_{k \in N(y) \cap V} eval(k) \right] \right) - b \\
&= b \sum_{y \in N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) - \sum_{y \in N(x)} \left(\left[\frac{1}{\nu_y} - \frac{1}{\nu_y + 1} \right] \sum_{k \in N(y) \cap V} eval(k) \right) \\
&= b \sum_{y \in N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) - \sum_{y \in N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right)
\end{aligned}$$

The result follows. □

The analogue of Proposition 5.1 in the AVKN context is [2, Prop. 8]. However, note that in the presence of non-voters, things are much more complicated: in [2, Prop. 8] the revenue obtained by some efficient bribing strategy is a function solely of the network structure and bribing strategy, whereas according to Proposition 5.1 the distribution of voters and the initial evaluation also play a role.

Consider now the revenue acquired from bribing a voter $x \in V$ with a bribe $\sigma(x) = b > 0$ (this is the unproven Proposition 12 from [2]).

Proposition 5.2. *Let $V \subseteq C$, let $x \in V$, and let σ be an efficient bribing strategy such that $\sigma(x) = b > 0$ and $\sigma(y) = 0$, for all $y \in C \setminus \{x\}$. For each $y \in N(x)$, set $\nu_y = |N(x) \cap V|$. The revenue gained is*

$$b \left[\Big(\sum_{y \in N(x)} \frac{1}{\nu_y} \Big) - 1 \right].$$

Proof. As in the proof of Proposition 5.1, we obtain

$$r_{\mathbb{P}}(\sigma) = \sum_{y \in N(x)} \left(\left[\frac{1}{\nu_y^\sigma} \sum_{k \in N(y) \cap V^\sigma} eval^\sigma(k) \right] - \left[\frac{1}{\nu_y} \sum_{k \in N(y) \cap V} eval(k) \right] \right) - b$$

where: $V^\sigma = V$; for every $y \in N(x)$, $\nu_y^\sigma = \nu_y$; and $eval^\sigma$ differs from $eval$ only on x, with $eval^\sigma(x) = eval(x) + b$. So

$$\begin{aligned} r_{\mathbb{P}}(\sigma) &= \sum_{y \in N(x)} \left(\left[\frac{1}{\nu_y} \sum_{k \in N(y) \cap V} eval^\sigma(k) \right] - \left[\frac{1}{\nu_y} \sum_{k \in N(y) \cap V} eval(k) \right] \right) - b \\ &= \sum_{y \in N(x)} \left(\left[\frac{1}{\nu_y} \sum_{k \in N(y) \cap V} eval(k) \right] + \frac{b}{\nu_y} - \left[\frac{1}{\nu_y} \sum_{k \in N(y) \cap V} eval(k) \right] \right) - b \\ &= \left(\sum_{y \in N(x)} \frac{b}{\nu_y} \right) - b = b \left[\left(\sum_{y \in N(x)} \frac{1}{\nu_y} \right) - 1 \right]. \end{aligned}$$

The result follows. □

Note that the revenue acquired by bribing a voter as described in Proposition 5.2 is dependent upon the network structure, the bribe, and the distribution of voters, though not on the initial evaluation as is the case for Proposition 5.1.

In the AVKN scenario of [2], the characterisation of the revenue obtained from some efficient bribing strategy in [2, Prop. 8] is used to develop a simple greedy algorithm that results in a weakly-dominant bribing strategy. Key to this greedy algorithm is the fact that the revenue obtained from an efficient bribing strategy is a function only of the network structure and the bribes. As such, it does not matter whether we bribe customer x before customer y or alternatively customer y before customer x. This is not so clear for us, given the involvement in the revenue computation of the distribution of voters and the initial evaluation. We now explore how the order of bribing customers impacts upon the eventual revenue.

Proposition 5.3. *Let $V \subseteq C$ give rise to an evaluation eval, let $x, x' \in C \setminus V$ be distinct non-voters, and let σ be an efficient strategy such that $\sigma(x) = b > 0$, $\sigma(x') = b' > 0$, and $\sigma(y) = 0$, for all $y \in C \setminus \{x, x'\}$. No matter whether we bribe x before x' or x' before x, the resulting cumulative revenue will be the same.*

Proof. Suppose that we bribe x first. So, by Proposition 5.1, the revenue r_1 acquired is

$$b \sum_{y \in N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) - \sum_{y \in N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right),$$

where for each $y \in N(x) \cup N(x')$, ν_y is defined as $|N(y) \cap V|$. Denote the evaluation obtained after bribing x by $eval'$ and the resulting set of voters by V'; in particular, $eval'$ differs from $eval$ only on x and $V' = V \cup \{x\}$. For each $y \in N(x')$, define ν'_y as $|N(y) \cap V'|$. So, again by Proposition 5.1, the additional revenue r_2 acquired by bribing x' is

$$b' \sum_{y \in N(x')} \left(\frac{1}{\nu'_y + 1} - \frac{1}{|N(x')|} \right) - \sum_{y \in N(x')} \left(\frac{1}{\nu'_y(\nu'_y + 1)} \sum_{k \in N(y) \cap V'} eval'(k) \right).$$

Note that any ν'_y depends upon whether $y \in N(x') \cap N(x)$ or whether $y \in N(x') \setminus N(x)$: in the former case, $\nu'_y = \nu_y + 1$; and in the latter case, $\nu'_y = \nu_y$. Consequently, we can rewrite r_2 as

$$\begin{aligned}
& b' \sum_{y \in N(x') \cap N(x)} \left(\frac{1}{\nu_y + 2} - \frac{1}{|N(x')|} \right) + b' \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x')|} \right) \\
& \quad - \sum_{y \in N(x') \cap N(x)} \left(\frac{1}{(\nu_y + 1)(\nu_y + 2)} \sum_{k \in N(y) \cap V'} eval'(k) \right) \\
& \quad - \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V'} eval'(k) \right) \\
= \; & b' \sum_{y \in N(x') \cap N(x)} \left(\frac{1}{\nu_y + 2} - \frac{1}{|N(x')|} \right) + b' \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x')|} \right) \\
& \quad - \sum_{y \in N(x') \cap N(x)} \left(\frac{1}{(\nu_y + 1)(\nu_y + 2)} \left(b + \sum_{k \in N(y) \cap V} eval(k) \right) \right) \\
& \quad - \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right).
\end{aligned}$$

Now we have that

$$\begin{aligned}
r_1 + r_2 &= b \sum_{y \in N(x) \cap N(x')} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) + b \sum_{y \in N(x) \setminus N(x')} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) \\
&\quad + b' \sum_{y \in N(x') \cap N(x)} \left(\frac{1}{\nu_y + 2} - \frac{1}{|N(x')|} \right) + b' \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x')|} \right) \\
&\quad - \sum_{y \in N(x) \cap N(x')} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right) \\
&\quad - \sum_{y \in N(x) \setminus N(x')} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right) \\
&\quad - \sum_{y \in N(x') \cap N(x)} \left(\frac{1}{(\nu_y + 1)(\nu_y + 2)} \left(b + \sum_{k \in N(y) \cap V} eval(k) \right) \right) \\
&\quad - \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right) \\
&= b \Bigg[\sum_{y \in N(x) \cap N(x')} \left(\frac{1}{\nu_y + 2} - \frac{1}{|N(x)|} \right) + \sum_{y \in N(x) \setminus N(x')} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) \Bigg] \\
&\quad + b' \Bigg[\sum_{y \in N(x') \cap N(x)} \left(\frac{1}{\nu_y + 2} - \frac{1}{|N(x')|} \right) + \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x')|} \right) \Bigg] \\
&\quad - \sum_{y \in N(x) \cap N(x')} \left(\frac{1}{\nu_y(\nu_y + 2)} \sum_{k \in N(y) \cap V} eval(k) \right) \\
&\quad - \sum_{y \in N(x) \setminus N(x')} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right) \\
&\quad - \sum_{y \in N(x') \setminus N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right).
\end{aligned}$$

By the symmetry of the expression for $r_1 + r_2$, were we to first bribe x' and then x, we would get exactly the same revenue. The result follows. $\square$

Now we consider the case where we compare bribing a non-voter x and then a voter x' with bribing the voter x' and then the non-voter x.

Proposition 5.4. *Let $V \subseteq C$ give rise to an evaluation eval, let $x \in C \setminus V$, let $x' \in V$, and let σ be an efficient strategy such that $\sigma(x) = b > 0$, $\sigma(x') = b' > 0$, and $\sigma(y) = 0$, for all $y \in C \setminus \{x, x'\}$. No matter whether we bribe x before x' or x' before x, the resulting cumulative revenue will be the same.*

Proof. For every $y \in N(x') \cup N(x)$, define ν_y as $|N(y) \cap V|$. Suppose that we first bribe the voter x' with b', to acquire revenue r_1, before bribing the non-voter x with b, to acquire revenue r_2. By Proposition 5.2, we have that

$$r_1 = b'\left[\left(\sum_{y \in N(x')} \frac{1}{\nu_y}\right) - 1\right].$$

Denote the evaluation obtained from *eval* by bribing x' by $eval'$; so, $eval'$ is identical to *eval* except on x' where $eval'(x') = eval(x') + b'$. Note also that the set of voters has not changed. By Proposition 5.1, we have that

$$r_2 = b \sum_{y \in N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|}\right) - \sum_{y \in N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval'(k)\right).$$

Hence, we have that $r_1 + r_2$ is equal to

$$\begin{aligned}
& b'\Bigg[\Bigg(\sum_{y\in N(x')}\frac{1}{\nu_y}\Bigg)-1\Bigg]+b\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y+1}-\frac{1}{|N(x)|}\Bigg) \\
& -\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\sum_{k\in N(y)\cap V} eval'(k)\Bigg) \\
= \; & b'\Bigg[\Bigg(\sum_{y\in N(x')}\frac{1}{\nu_y}\Bigg)-1\Bigg]+b\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y+1}-\frac{1}{|N(x)|}\Bigg) \\
& -\sum_{y\in N(x)\cap N(x')}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\sum_{k\in N(y)\cap V} eval'(k)\Bigg) \\
& -\sum_{y\in N(x)\setminus N(x')}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\sum_{k\in N(y)\cap V} eval'(k)\Bigg) \\
= \; & b'\Bigg[\Bigg(\sum_{y\in N(x')}\frac{1}{\nu_y}\Bigg)-1\Bigg]+b\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y+1}-\frac{1}{|N(x)|}\Bigg) \\
& -\sum_{y\in N(x)\cap N(x')}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\Bigg(b'+\sum_{k\in N(y)\cap V} eval(k)\Bigg)\Bigg) \\
& -\sum_{y\in N(x)\setminus N(x')}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\sum_{k\in N(y)\cap V} eval(k)\Bigg) \\
= \; & b'\Bigg[\sum_{y\in N(x')}\Bigg(\frac{1}{\nu_y}\Bigg)-\sum_{y\in N(x)\cap N(x')}\Bigg(\frac{1}{\nu_y}-\frac{1}{\nu_y+1}\Bigg)-1\Bigg] \\
& +b\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y+1}-\frac{1}{|N(x)|}\Bigg) \\
& -\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\sum_{k\in N(y)\cap V} eval(k)\Bigg) \\
= \; & b'\Bigg[\sum_{y\in N(x')\cap N(x)}\Bigg(\frac{1}{\nu_y+1}\Bigg)+\sum_{y\in N(x')\setminus N(x))}\Bigg(\frac{1}{\nu_y}\Bigg)-1\Bigg] \\
& +b\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y+1}-\frac{1}{|N(x)|}\Bigg) \\
& -\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\sum_{k\in N(y)\cap V} eval(k)\Bigg).
\end{aligned}$$

Now suppose that we first bribe the non-voter x with b, to acquire revenue s_1, before bribing the voter x' with b', to acquire revenue s_2. By Proposition 5.1, we have that

$$s_1 = b\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y+1}-\frac{1}{|N(x)|}\Bigg)-\sum_{y\in N(x)}\Bigg(\frac{1}{\nu_y(\nu_y+1)}\sum_{k\in N(y)\cap V} eval(k)\Bigg).$$

Denote the evaluation obtained from *eval* by bribing x by $eval'$, denote the set of voters after bribing x by V', and for every $y \in N(x')$, set ν'_y as $|N(y) \cap V'|$; so, $eval'$ is identical to *eval* except on x, where $eval'(x) = b$, and $V' = V \cup \{x\}$. By Proposition 5.2, we have that

$$s_2 = b'\Bigg[\Bigg(\sum_{y \in N(x')} \frac{1}{\nu'_y}\Bigg) - 1\Bigg].$$

Hence, we have that $s_1 + s_2$ is equal to

$$\begin{aligned}
& b \sum_{y \in N(x)} \Bigg(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|}\Bigg) - \sum_{y \in N(x)} \Bigg(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k)\Bigg) \\
& + b'\Bigg[\Bigg(\sum_{y \in N(x')} \frac{1}{\nu'_y}\Bigg) - 1\Bigg] \\
= \; & b \sum_{y \in N(x)} \Bigg(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|}\Bigg) - \sum_{y \in N(x)} \Bigg(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k)\Bigg) \\
& + b'\Bigg[\Bigg(\sum_{y \in N(x') \cap N(x)} \frac{1}{\nu_y + 1}\Bigg) + \Bigg(\sum_{y \in N(x') \setminus N(x)} \frac{1}{\nu_y}\Bigg) - 1\Bigg]
\end{aligned}$$

Hence, we have that $r_1 + r_2 = s_1 + s_2$ and the result follows. □

Exactly what Propositions 5.3 and 5.4 show is that in fact it is not necessary to define an ordering in which a strategy over both voters and non-voters should be evaluated.

5.3 Computation of Bribing Strategies

We now develop a greedy efficient bribing strategy, based upon Propositions 5.1 - 5.4. From Propositions 5.1 - 5.4, the order of bribing in some efficient strategy does not matter and without loss of generality we may assume that every customer is bribed at most once (otherwise, manipulate the order of bribing so that some customer is bribed in consecutive bribes and combine the two bribes into one).

Suppose that we are given some network (C, E) and some evaluation *eval* (so that every customer is adjacent to at least one voter). Suppose also that we consider bribing some customer $x \in C$ with the value $b > 0$; so, $eval(x) + b \leq 1$. The revenue we would acquire by doing so, which we denote $r_{\mathbb{P}}(x, b, eval)$, is given by Proposition 5.1, if x is a non-voter, and by Proposition 5.2, if x is a voter.

Our greedy algorithm is as follows.

```
Data: Evaluation function $eval$ and network $(C, E)$.
Result: An efficient bribing strategy $\sigma : C \to Val$; the total bribe made; and the
        total revenue acquired.
compute $budget = u^0_{\mathbb{P}}$;
set $r^\sigma_{\mathbb{P}} = 0$ and $bribe^\sigma = 0$;
while $budget > 0$ do
    for each $y \in C$, let $b_y = \min\{1 - eval(y), budget\}$;
    let $x \in C$ be such that $r_{\mathbb{P}}(x, b_x, eval) \geq r_{\mathbb{P}}(y, b_y, eval)$, for all $y \in C$;
    if $r_{\mathbb{P}}(x, b_x, eval) > 0$ then
        bribe $x$ with $b_x$ and amend $eval$ accordingly;
        $budget = budget - b_x$;
        $bribe^\sigma = bribe^\sigma + b_x$;
        $r^\sigma_{\mathbb{P}} = r^\sigma_{\mathbb{P}} + r_{\mathbb{P}}(x, b_x, eval)$
    else
        $budget = 0$
    end
end
return $(\sigma, bribe^\sigma, r^\sigma_{\mathbb{P}})$
```

Algorithm 2: A greedy efficient bribing strategy in the NVKL scenario.

The algorithm recomputes, at each stage, the projected revenue of bribing each customer the maximal efficient amount. The customer whose projected revenue is highest is bribed by that amount. This process continues until we run out of utility to reinvest. Upon each iteration of this algorithm, we must recompute new projected revenues. Although these projected revenues are not static (unlike the case of AVKL), the algorithm clearly still runs in polynomial-time.

Note that up until now we have been working in exactly the same framework as [2] as regards our set *Val*. Just as with the greedy algorithm (Algorithm 1) from [2], our greedy algorithm from NVKL (Algorithm 2) can make arbitrary (rational) bribes from $[0, 1]$ to customers irregardless of the set *Val* from which the customer evaluations are drawn (which is understandable as no matter what the set *Val* is, the $\mathbb{P}$-ratings of customers can be rational numbers). For the rest of this section, and in particular for our **NP**-completeness proof, we assume that customers choose their evaluations from the set of rational numbers in $[0, 1]$. We shall comment on this assumption when we discuss future work.

5.3.1 Failure of the Greedy Approach

Unfortunately we shall show that the greedy algorithm presented above fails to find a weakly-dominant strategy in general. Although we have established that for some efficient bribe amounts $b_1, b_2 > 0$ to any two customers of C, the order in which we execute these bribes does not change the resulting revenue, the difficulty arises in choosing these amounts and therefore the customers whom we should bribe. Given the existence of non-voters, the optimal choice might even be to bribe some non-voter an amount so as to *lose* utility. The reason for this is that this now modifies the structure of the network in that turning a non-voter into a voter affects the $\mathbb{P}$-rating and the projected revenues of all the customers in its neighbourhood. The network now might be such that there are even more profitable bribes to be made, even given the prior loss that was made. We will eventually see that it cannot possibly be that the (polynomial-time) greedy algorithm yields a weakly-dominant strategy due to the **NP**-completeness of the problem we are attempting to solve (unless, of course, **P** = **NP**).

We present the following counter-example to the optimality of the greedy algorithm.

Example 5.1. *Consider a* 6*-clique* $\mathcal{X}$ *of non-voters, each connected to a unique voter of evaluation* $\frac{1}{2}$ *as is depicted in Figure 5.1.*

We begin to execute the greedy algorithm on the network. We calculate the initial utility of the network as follows

$$u^0_{\mathbb{P}} = \sum_{c \in C} \mathbb{P}\text{-rating}(c, eval) = \sum_{c \in \mathcal{X}} \frac{1}{2} + \sum_{c \in C \setminus \mathcal{X}} \frac{1}{2} = 6.$$

Suppose that we bribe some clique customer $x \in \mathcal{X}$ its maximal amount. The revenue gain from doing so, by Proposition 5.1, is

$$\sum_{y \in N(x)} \left(\frac{1}{\nu_y + 1} - \frac{1}{|N(x)|} \right) - \sum_{y \in N(x)} \left(\frac{1}{\nu_y(\nu_y + 1)} \sum_{k \in N(y) \cap V} eval(k) \right)$$

$$= \sum_{y \in N(x)} \left(\frac{1}{2} - \frac{1}{7} \right) - \sum_{y \in N(x)} \left(\frac{1}{2} \sum_{k \in N(y) \cap V} \frac{1}{2} \right) = \frac{3}{4}.$$

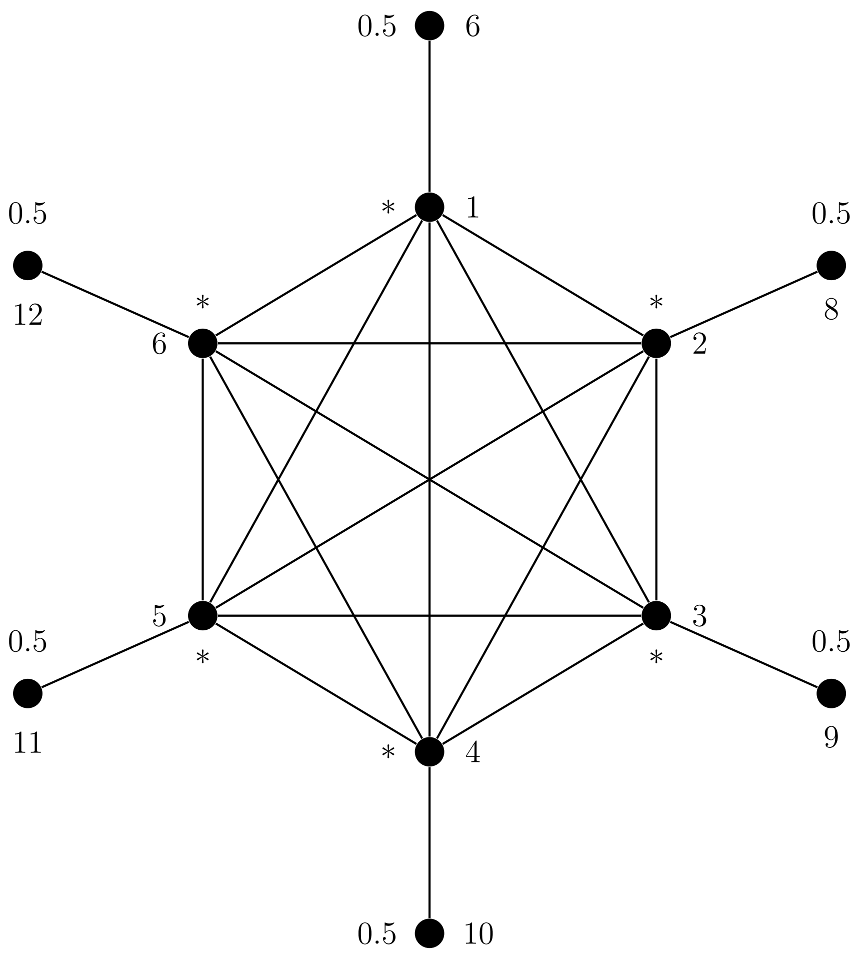

Figure 5.1: 6-clique of non-voters each attached to a voter.

Alternatively, suppose we bribe some non-clique customer $x \in C \setminus \mathcal{X}$ its maximal amount $\frac{1}{2}$. The revenue gain from doing so, by Propoistion 5.2, is

$$\frac{1}{2}\left[\left(\sum_{y\in N(x)} \frac{1}{\nu_y}\right) - 1\right] = \frac{1}{2}\left[\left(\sum_{y\in N(x)} 1\right) - 1\right] = \frac{1}{2}.$$

Since $\frac{3}{4} > \frac{1}{2}$, Algorithm 2 first bribes a clique-customer the amount 1.

Consider now, any currently unbribed non-voter and the pair it makes with the unique voter it is adjacent to, as is depicted in Figure 5.2 by the vertices coloured in red. As long as the non-voter remains unbribed, we can bribe the voter by $\frac{1}{2}$ and gain an increase in revenue. Consequently, Algorithm 2 must bribe at least one customer of the pair. This

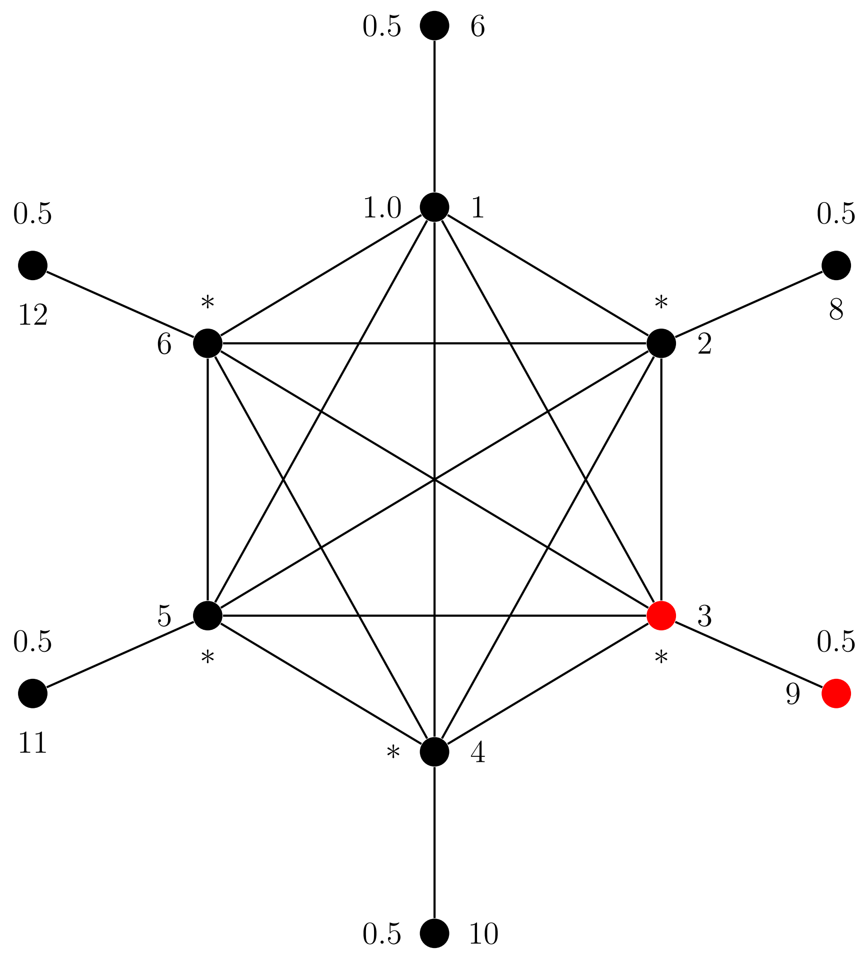

Figure 5.2: Customer 1 is bribed. We consider the voter/non-voter pair made up of customers 3 and 9.

is true for all five remaining such pairs. Therefore, the least amount that Algorithm 2 bribes, from here on, is $\frac{1}{2}$ per pair.

The revenue produced by the bribing strategy computed by Algorithm 2 is at most

$$r_{\mathbb{P}}(\sigma) \leq 12 - 6 - 1 - \frac{5}{2} = \frac{5}{2}.$$

This is due to the fact that the maximum utility of the network after executing any strategy σ is $12 = |C|$ (by Proposition 6.1), the initial utility of the network is 6, the amount 1 is spent on the first bribe, and at least $\frac{1}{2}$ is spent on bribing the remaining five voter/non-voter pairs (we established that at least one customer of each of these pairs must be bribed).

Consider the strategy σ' of bribing all non-clique customers fully. That is,

$$\sigma'(6) = \sigma'(7) = \sigma'(8) = \sigma'(9) = \sigma'(10) = \sigma'(11) = \sigma'(12) = \frac{1}{2}$$

and $\sigma'(x) = 0$ for all other customers x. The revenue gained by playing this strategy is

$$r_{\mathbb{P}}(\sigma') = \sum_{c \in C} \mathbb{P}\text{-rating}(c, eval) - u^0_{\mathbb{P}} - 3.$$

The sole voter adjacent to every customer of C is fully bribed (has an evaluation of 1). Therefore, for all customers of the network, the average evaluation of all voting neighbours is 1; that is, the sum of all $\mathbb{P}$-ratings is 12. This strategy yields a revenue of $r_{\mathbb{P}}(\sigma') = 12 - 6 - 3 = 3$ and clearly this is greater than the maximal revenue that Algorithm 2 could possibly obtain. We therefore conclude that the greedy algorithm does not compute a weakly-dominant bribing strategy in all cases.

Note that we could have applied Algorithm 2 in order to obtain the exact bribing strategy that the algorithm would compute. However, this computation is often tedious to do by hand, particularly for networks such as that of Figure 5.1. All we have done is show that after the greedy algorithm makes even just its first choice, the maximum revenue that it could possibly obtain from that point onwards in bounded from above. Furthermore, we have shown there exist bribing strategies that we can see 'by eye', that yield greater revenues than this.

The question now, is whether we can find a weakly-dominant bribing strategy in polynomial-time or whether there might be a complexity-theoretic barrier to doing so. In the next chapter, we now show that the latter case holds.

5.4 NVKL is NP-Complete

In order to prove a complexity-theoretic hardness result for the problem of computing a weakly-dominant bribing strategies under NVKL, we must first precisely formulate the problem as a decision problem. Let us define the decision problem NVKL as follows.

Definition 5.1. *NVKL*

- *An instance is a finite network (C, E), an initial evaluation $eval_0$, and $\rho \in \mathbb{Q}$.*
- *A yes-instance is an instance of NVKL such that there exists a bribing strategy σ such that the resulting revenue is at least ρ.*

Clearly, any instance of the above problem should adhere to the usual restrictions of the framework. These are, most importantly, that the initial evaluation is such that every customer $c \in C$ is adjacent to at least one customer $c' \in C$ such that $eval(c') \neq *$ (recall that every customer is adjacent to itself). Also, any strategy σ is such that $\sum_{c \in C} \sigma(c)$ is at most the initial utility resulting from $eval_0$.

We claim that we can construct a polynomial-time reduction from the independent set problem on 3-regular graphs. This problem is **NP**-complete [19]. The problem is defined as follows. Firstly, we recall the definition of an independent set.

Definition 5.2. *Independent Set: A graph G has an independent set of size k if there exists a set X of k vertices such that there is no edge joining any pair of vertices of X.*

Definition 5.3. *ISREG(3)*

- *An instance is a pair (G, k) where G is a 3-regular graph and $k \in \mathbb{N}$.*
- *A yes-instance is an instance such that G has an independent set of size k.*

5.4.1 Proof Overview

In order to show that NVKL is **NP**-complete, we proceed by reduction from the problem ISREG(3). Given an instance (G, k) of ISREG(3), we construct a customer network (C, E), define an initial evaluation $eval_0$, and define an amount ρ. Our customers are partitioned into old customers, who correspond to the vertices of G, and additional new customers. Firstly, we assume that (G, k) is a yes-instance of ISREG(3) and show that bribing the k customers (in the newly constructed network) which correspond to the vertices that form the independent set in G yields a revenue of ρ. We conclude that $((C, E), eval_0, \rho)$ is therefore a yes-instance of NVKL.

Next, we suppose that we are given some yes-instance $((C, E), eval_0, \rho)$ of NVKL. We firstly prove Lemma 5.1 which states that we are able to move amounts of bribe between customers of the network (adhering to certain restrictions that will be explained). Making use of this lemma, we are able to show that without loss of generality we may assume that σ is such that only old customers are bribed, all old customers are fully bribed, and exactly k old customers are bribed. We obtain this result by starting with an arbitrary weakly-dominant strategy and obtaining a series of suppositions and contradictions. Having done this, we consider the positions in the customer network of the bribed old customers. We are able to show that without loss of generality, these customers are positioned such that the vertices in the initial instance of ISREG(3) that they correspond to form an independent set in G.

The end result is that we have shown

- Given an arbitrary instance (G, k) of ISREG(3), we can construct an instance $((C, E), eval_0, \rho)$ of NVKL so that
  - (G, k) is a yes-instance of ISREG(3) $\implies$ $((C, E), eval_0, \rho)$ is a yes-instance of NVKL.
  - $((C, E), eval_0, \rho)$ is a yes-instance of NVKL $\implies$ (G, k) is a yes-instance of ISREG(3).

Finally, it should be noted that NVKL is clearly in **NP**. This is obvious from the fact that given an instance $((C, E), eval_0, \rho)$ we can check in polynomial-time that an arbitrary bribing strategy σ yields a revenue of at least ρ in (C, E): all that is to be done is sum the $\mathbb{P}$-ratings of all customers of C. After having shown the above result we are therefore able to conclude that NVKL is **NP**-complete.

5.4.2 A Polynomial-Time Reduction from ISREG(3) to NVKL

Given any instance of ISREG(3) we can build a network as follows.

- Let us begin by setting the set C of customers of our instance of NVKL to be the set of vertices of G. We call these customers the 'old' customers.
- For all $v \in G$, let us introduce n new customers $v_1, .., v_n \in C$ where n is the number of vertices of G. Also we add $(v, v_i) \in E$ for $i = 1, 2, ..., n$. We refer to these new customers as the new *pendant customers.*
- For each edge (u, v) of G, we introduce a new customer $w_{u,v}$, with these customers called the new *edge customers.*
- For every edge (u, v) of G, we add $(u, w_{u,v})$ and $(w_{u,v}, v)$ to E.

The construction of a network of customers from the graph G (of our instance of IS-REG(3)) can be visualised as follows in Figure 5.4.

For any such network as constructed above, we can define $eval_0$ as follows, where $0 < \epsilon < 1$ is some value to be chosen momentarily.

- If $c \in C$ is an old customer then c is chosen to be a non-voter.
- If $c \in C$ is a new edge or new pendant customer then $eval_0(c) = \epsilon$.

By the construction of the network, we have that for all $c \in C$, the $\mathbb{P}$-rating$(c, eval_0) = \epsilon$. Every customer of the newly constructed customer network contributes ϵ to the initial utility of the network and therefore $u^0_{\mathbb{P}} = \epsilon(n + n^2 + \frac{3n}{2})$. We now choose ϵ so that $u^0_{\mathbb{P}} = k$; that is, so that

$$\epsilon = \frac{k}{n + n^2 + \frac{3n}{2}}.$$

By assumption the restaurant owner can only make bribes totalling at most k. Furthermore, note that the initial evaluation is a valid one in that every customer of the network is adjacent to at least one voter. Finally, let

$$\rho = k(1 - \epsilon)\left(\frac{1}{n + 4} + \frac{n + 3}{2}\right).$$

We shall explain why we chose ϵ and ρ as we do later.

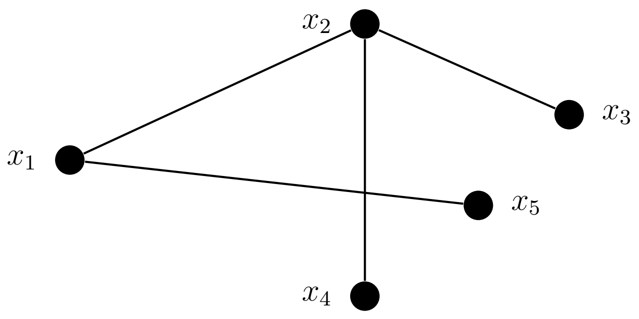

Figure 5.3: Segment of 3-regular graph G.

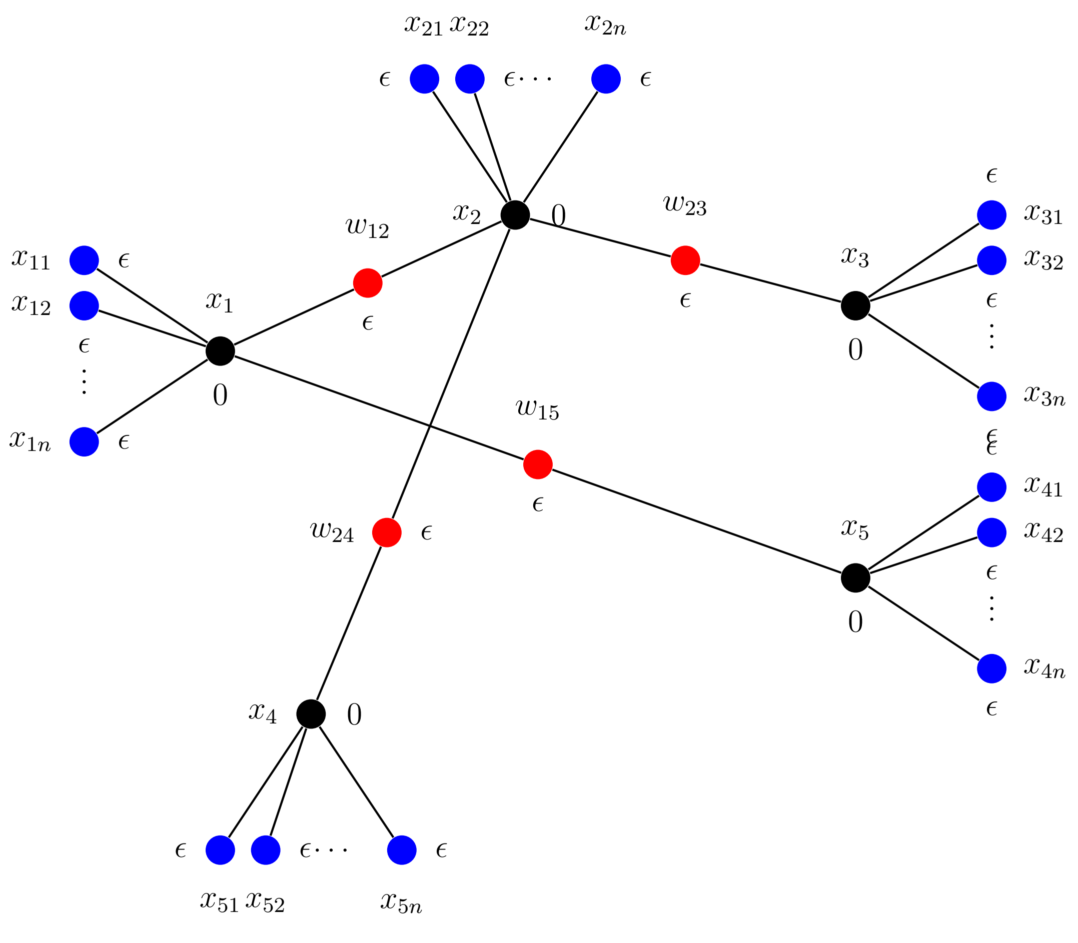

Figure 5.4: Network (C, E) constructed from G.

Suppose that our instance (G, k) of ISREG(3) is a yes-instance; that is, there is a set I of k vertices such that no two vertices of I are adjacent in G. Consider the bribing strategy for (C, E) (as constructed above) where $\sigma(c) = 1$, for every old customer corresponding to some vertex of I, and $\sigma(c') = 0$ for all other $c' \in C$. The resulting revenue is given by the increase in $\mathbb{P}$-rating of the bribed customers and their neighbourhoods. The cumulative increase in utility due to these bribed customers is the sum of the revenues of old customers, new pendant customers, and new edge customers. The cumulative increase due to the new pendant vertices adjacent to bribed old customers is

$$k\left(\frac{1+(n+3)\epsilon}{n+4}-\epsilon\right)=k\frac{1-\epsilon}{n+4},$$

the cumulative increase due to the new pendant customers adjacent to old bribed customers is

$$nk\left(\frac{1+\epsilon}{2}-\epsilon\right)=nk\frac{1-\epsilon}{2},$$

and the cumulative increase in revenue due to the new edge customers adjacent to bribed old customers is

$$3k\left(\frac{1+\epsilon}{2}-\epsilon\right)=3k\frac{1-\epsilon}{2}$$

(note that this is because the bribed old customers correspond to an independent set in G). Hence, the total revenue accruing due to σ is

$$k(1-\epsilon)\left(\frac{1}{n+4}+\frac{n+3}{2}\right)-k=\rho.$$

Therefore $((C,E),eval_0,\rho)$ is a yes-instance of NVKL.

Remark 5.1. *We can see now why we chose ρ as we did. It is precisely the revenue accrued when we bribe every old customer corresponding to the vertices of an independent set of size k in G.*

Conversely, we now suppose that $((C,E),eval_0,\rho)$ is a yes-instance of NVKL and that σ is a bribing strategy that yields a revenue of at least ρ. We will show that σ can be manipulated so that without loss of generality we may assume that our optimal bribing strategy σ is such that only old customers are bribed, all bribed old customers are bribed fully, and exactly k old customers are bribed.

Lemma 5.1. *Let σ be an optimal bribing strategy. Let X be the set of customers for which $eval(x)<eval^{\sigma}(x)<1$. Let $v_y=|N(y)\cap V|$ for any $y\in C$. For $x,y\in X$*

$$\sum_{z\in N(x)}\frac{1}{v_z}=\sum_{z\in N(y)}\frac{1}{v_z}.$$

Proof. We consider the effect of transferring $\delta>0$ of the bribe of x to the bribe of y, where δ is as small as we like and where $eval(x)<eval^{\sigma}(x)-\delta<eval^{\sigma}(x)+\delta<1$ and

$eval(y) < eval^\sigma(y) - \delta < eval^\sigma(y) + \delta < 1$. By Proposition 5.2, we have that the utility change is as follows.

$$\delta\left(\sum_{z\in N(y)} \frac{1}{v_z}\right) - \delta\left(\sum_{z\in N(x)} \frac{1}{v_z}\right)$$

$$= \delta\left(\sum_{z\in N(y)} \frac{1}{v_z} - \sum_{z\in N(x)} \frac{1}{v_z}\right)$$

Since we know that σ is an optimal bribing strategy, this change cannot yield a positive change in network utility. Thus, we can bound this quantity as follows.

$$\delta\left(\sum_{z\in N(y)} \frac{1}{v_z} - \sum_{z\in N(x)} \frac{1}{v_z}\right) \leq 0$$

Now, we consider the effect of transferring δ amount of the bribe of y to the bribe of x. The change in network utility is given by the following.

$$\delta\left(\sum_{z\in N(x)} \frac{1}{v_z}\right) - \delta\left(\sum_{z\in N(y)} \frac{1}{v_z}\right)$$

$$= \delta\left(\sum_{z\in N(x)} \frac{1}{v_z} - \sum_{z\in N(y)} \frac{1}{v_z}\right)$$

With similar reasoning to the above, we have that

$$\delta\left(\sum_{z\in N(x)} \frac{1}{v_z} - \sum_{z\in N(y)} \frac{1}{v_z}\right) \leq 0.$$

Combining these two inequalities yields

$$\sum_{z\in N(x)} \frac{1}{v_z} \leq \sum_{z\in N(y)} \frac{1}{v_z},$$

and

$$\sum_{z\in N(y)} \frac{1}{v_z} \leq \sum_{z\in N(x)} \frac{1}{v_z},$$

which finally imply that

$$\sum_{z\in N(y)} \frac{1}{v_z} = \sum_{z\in N(x)} \frac{1}{v_z}.$$

□

Remark 5.2. *Lemma 5.1 says that, given an optimal bribing strategy, we can move bribes amongst non-fully bribed voters arbitrarily without decreasing nor increasing the revenue acquired so long as*

- *we do not totally remove all the bribe from a bribed customer that was not originally a non-voter*
- *we do not turn a non-voting customer into a voting one by introducing a bribe.*

The above applies to any customer network with some given initial evaluation and some optimal bribing strategy. Consider applying the above to the customer network (C, E) constructed from the initial instance of ISREG(3). Suppose that we have the following profile for customers bribed by σ

- if an old customer is bribed then the bribes on old customers, in descending order, are $1, 1, ..., 1, \delta_1, \delta_2, ..., \delta_m$, for some $m \geq 0$ and where $0 < \delta_i < 1$, for each $i = 1, 2, ..., m$, with possibly no bribes of 1
- if a new customer is bribed then the bribes, in descending order, are $1-\epsilon, 1-\epsilon, ..., 1-\epsilon, \epsilon_1, \epsilon_2, ..., \epsilon_s$ for some $s \geq 0$ and where $0 < \epsilon_i < 1-\epsilon$ for each $i = 1, 2, ..., s$, with possibly no bribes of $1-\epsilon$ (recall that the initial evaluation of any new customer is $\epsilon > 0$).

Without loss of generality, we may assume that $\sum_{i=1}^{m} \delta_i \leq 1$; otherwise, we would have that $m \geq 2$ and we could reduce the bribes $\delta_2, \delta_3, ..., \delta_m$, without making any equal to zero, so as to increase the bribe δ_1 to 1 and secure another fully bribed customer.

Firstly, we introduce the following lemma which we will later make use of.

Lemma 5.2. *Let (C, E) be some network with initial evaluation $eval_0$ and where bribes have been made so that the evaluation is now $eval$. Suppose that $c \in C$ was initially a non-voter and is now such that $eval(c) = \delta > 0$ but where for every voting or bribed customer c'' of $\bigcup\{N(c') : x' \in N(c)\}$, we have that $\delta < eval(c'')$. Then, amending the evaluation so that the bribe is removed from c results in an increase in revenue.*

Proof. By definition we have that

$$u^0_{\mathbb{P}} = \sum_{z \in C} \mathbb{P}\text{-rating}(z, eval).$$

For all $z \in N^2(x)$, we know that $eval(z) > \delta$. In order to examine the change in utility after setting $eval(x) = *$, we should consider the $\mathbb{P}$-ratings of the neighbours of x. Let y be a neighbour of x. The average $\mathbb{P}$-rating of its voting neighbours is clearly higher when $eval(x) = *$, since the previous evaluation of x (δ) is lower than the evaluation of all of its other neighbours. Therefore, all $\mathbb{P}$-ratings of all neighbours of x *increase*; that is, the utility also increases. □

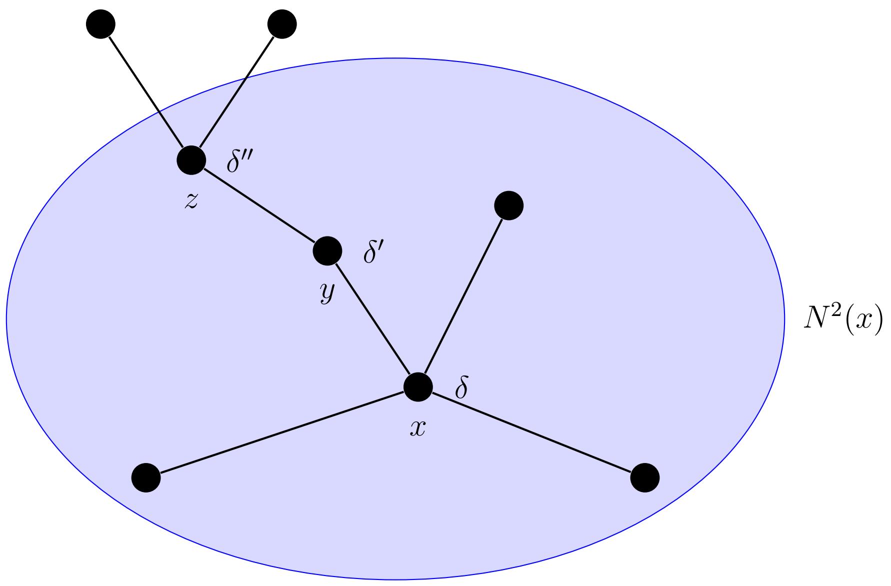

Figure 5.5: The second neighbourhood of a voter x whose evaluation is decreased from δ to $*$. The second neighbourhood of x is shaded in blue.

Example 5.2. *For example, we see this in Figure 5.5, where we have $eval(x) = \delta$ being removed. We consider the $\mathbb{P}$-rating of some neighbour of x, say, y. $\mathbb{P}$-rating$(y, eval) = \frac{1}{2}(\delta' + \delta'')$ which is clearly higher than the original $\mathbb{P}$-rating of $\frac{1}{3}(\delta + \delta' + \delta'')$.*

We begin by assuming that $m > 1$ and then show that this leads to a contradiction. First, we make the smallest bribe δ_m so that $\delta_m < \delta_{m-1}$ and $\delta_m < \epsilon$ (by sharing some of the δ_m bribe amongst the $\delta_1, ..., \delta_{m-1}$ bribes; we note here that we are applying the reasoning subsequent to Lemma 5.1). Consider the old customer accepting the δ_1 bribe. The contribution of the revenue from this δ_1 bribe to c comes about due to the $\mathbb{P}$-ratings of c and its adjacent new customers; that is, the $\mathbb{P}$-ratings of the customers in $N(c)$ (which depend on the evaluations in $N^2(x)$). Suppose that we define a new bribing strategy σ' that is identical to σ except that no bribe is made to c any more (the evaluation of c is changed to *). By Lemma 5.2 we have generated more revenue by using the bribing strategy σ' which contradicts the optimally of σ. Consequently, we must have that $m \leq 1$; that is, we have at most one old customer that is not fully bribed.

Now we consider the new customers. We can move bribes amongst the new customers so that we may assume that all but at most one new customer is not fully bribed; that is, that $s \leq 1$. Recall that we may move all the bribe to a new customer c that is not fully bribed as this does not change the number of voters in the neighbourhood of c nor in the neighbourhood of any old customer adjacent to c. (See Remark 5.2)

Lemma 5.3. *There exists an old customer c' who has not been bribed and where at most one of its adjacent new pendant customers has been bribed.*

Proof. Our initial choice of ϵ was such that

$$\epsilon = \frac{k}{n + n^2 + \frac{3n}{2}}.$$

Furthermore, the amount invested k, is certainly less than n. Therefore we have that

$$\epsilon \leq \frac{n}{n + n^2 + \frac{3n}{2}} = \frac{2}{2n+5},$$

and

$$1 - \epsilon > \frac{2n+3}{2n+5}.$$

Let p be the number of old customers that have been fully bribed and let t be the number of new customers that have been bribed. By summing the total amount bribed, we have that $p + 1 + t(1 - \epsilon) < k$. Thus

$$t < \frac{k - p - 1}{1 - \epsilon} < \frac{(2n+5)(n-p-1)}{2n+3},$$

and therefore $t < 2(n - (p + 1))$. This says that the number of bribed new customers is strictly less than twice the number of old customers that remain unbribed. If every unbribed old customer was adjacent to two or more bribed new pendant customers then we would have that $t \geq 2(n - (p + 1))$ which is clearly not possible. Therefore we can conclude that there exists an unbribed old customer c' such that at most one of its adjacent new pendant vertices has been bribed. □

We now suppose that there exists a new customer that has been fully bribed and show that this yields a contradiction. Consider the bribe of $1 - \epsilon$ to c. We say that its contribution to the overall utility is the amount of utility that this $1 - \epsilon$ bribe generates due to its contribution towards the $\mathbb{P}$-ratings of c and c's adjacent old customers. If c is a new pendant customer then this contribution is certainly less than 2 as $|N(c)| = 2$, and if c is a new edge customer then this contribution is less than 3 as $|N(c)| = 3$. Therefore, in all cases, the bribe of $1 - \epsilon$ to c contributes less than 3 units to the overall utility accrued from σ.

By Lemma 5.3, let c' be an old customer that is not bribed and that is adjacent to at most one new pendant customer that has been bribed. Consider moving the $1 - \epsilon$ bribe from c to c'; so, we obtain a new (efficient) strategy σ'. Let us examine how much this utility this $1 - \epsilon$ bribe contributes, under σ', to the overall utility.

At least $n - 1$ of the new pendant customers adjacent to c' have not been bribed and so the cumulative increase in utility as a result of these new pendant customers is the sum of the new $\mathbb{P}$-ratings of the neighbours of c' minus the sum of the old $\mathbb{P}$-ratings. This is given by $(n-1)\frac{1}{2} - (n-1)\epsilon$ and given that $\epsilon \leq \frac{2}{2n+5}$ then the cumulative increase in utility is

$$(n-1)\left(\frac{1}{2} - \epsilon\right) > \frac{n-1}{2} - 1.$$

Bribing c' might reduce the $\mathbb{P}$-ratings of c' and its adjacent new edge customers. However, this reduction is certainly less than 4 units. Therefore we may conclude that the movement of $1 - \epsilon$ of bribe from c to c' increases the overall utility by an amount greater than

$$\left(\frac{n-1}{2} - 1\right) - 7$$

units. This amount is strictly positive for n sufficiently large ($n \geq 14$). Therefore the strategy σ' that we have constructed yields a revenue greater than that of σ. This contradicts the optimality of σ. Hence, we must have that no new customer has been fully bribed and so at most one new customer has been bribed. Furthermore, if one new customer has been bribed, then it has been bribed an amount less than $1-\epsilon$.

At this stage of the proof, we may assume, without loss of generality, that the bribing strategy σ is such that

- at most one old customer has not been fully bribed
- at most one new customer has been bribed (and if so has not been fully bribed).

Suppose that some new customer c has indeed been bribed some amount δ such that $0 < \delta < 1-\epsilon$. Again, we will show that this yields a contradiction. Without loss of generality, we may assume that all old customers have been fully bribed. To see this, assume the converse. We can either move (some of) the bribe from c to the old customer who has not been fully bribed, so as to fully bribe the old customer; or we can move the bribe from c to the old customer in its entirety, and we would not have a new customer that has been bribed which contradicts our assumption.

We shall examine the various cases that arise and obtain a contradiction to our hypothesis (that a new customer has been bribed but not fully). A key point to note throughout is that the total bribe made is strictly less than k (as it cannot be a whole number; recall, we may assume all old customers have been fully bribed) and so we have some additional bribe to spend if we wish.

Case 1: c is a new pendant customer and is adjacent to an old customer c' that has not been bribed.

Let us make an additional bribe of δ' to c (so that the strategy remains efficient and we do not over-extend our bribing resource; we can choose δ' to be as small as we like). The additional revenue that we obtain is due to the increase in the $\mathbb{P}$-ratings of c and c' minus the bribe made.

This amounts to

$$[(\epsilon+\delta+\delta')-(\epsilon+\delta)]+[\frac{((n+3)\epsilon+\delta+\delta')}{n+3}-\frac{((n+3)\epsilon+\delta)}{n+3}]-\delta'=\frac{\delta'}{n+3}$$

and is strictly positive. This contradicts the optimality of σ.

Case 2: c is a new pendant customer and is adjacent to an old customer c' that has been bribed.

Let us decrease the bribe of δ to c by the amount δ' (so that we do not remove the bribe altogether). The additional revenue that we obtain is due to the savings we make by only bribing c with the amount $\delta - \delta'$ minus the decrease in the $\mathbb{P}$-ratings of c and c'.

This amounts to

$$\begin{aligned}
&\delta' - [\frac{(1+\epsilon+\delta)}{2} - \frac{(1+\epsilon+\delta-\delta')}{2}] - [\frac{(1+(n+3)\epsilon+\delta)}{n+4} - \frac{(1+(n+3)\epsilon+\delta-\delta')}{n+4}] \\
= \quad &\delta' - \frac{\delta'}{2} - \frac{\delta'}{n+4} = \delta'(\frac{1}{2} - \frac{1}{n+4})
\end{aligned}$$

and is strictly positive. This contradicts the optimality of σ.

Case 3: c is a new edge customer and is adjacent to old customers neither of which has been bribed.

Let us make an additional bribe of δ' to c. Similarly to above, the additional revenue that we obtain amounts to

$$[(\epsilon+\delta+\delta') - (\epsilon+\delta)] + 2[\frac{((n+3)\epsilon+\delta+\delta')}{n+3} - \frac{((n+3)\epsilon+\delta)}{n+3}] - \delta' = \frac{2\delta'}{n+3}$$

and is strictly positive. This contradicts the optimality of σ.

Case 4: c is a new edge customer and is adjacent to old customers one of which has been bribed.

Let us decrease the bribe of δ to c by the amount δ'. Similarly to above, the additional revenue that we obtain amounts to

$$\begin{aligned}
&\delta' - [\frac{(1+\epsilon+\delta)}{2} - \frac{(1+\epsilon+\delta-\delta')}{2}] - [\frac{(1+(n+3)\epsilon+\delta)}{n+4} - \frac{(1+(n+3)\epsilon+\delta-\delta')}{n+4}] \\
&\quad - [\frac{((n+3)\epsilon+\delta)}{n+3} - \frac{((n+3)\epsilon+\delta-\delta')}{n+3}] \\
= \quad &\delta' - \frac{\delta'}{2} - \frac{\delta'}{n+4} - \frac{\delta'}{n+3} = \delta'(\frac{1}{2} - \frac{1}{n+4} - \frac{1}{n+3})
\end{aligned}$$

and is strictly positive. This contradicts the optimality of σ.

Case 5: c is a new edge customer and is adjacent to old customers both of which have been bribed.

Let us decrease the bribe of δ to c by the amount δ'. Similarly to above, the additional revenue that we obtain amounts to

$$\begin{aligned}&\delta' - [\frac{(2+\epsilon+\delta)}{2} - \frac{(2+\epsilon+\delta-\delta')}{2}] - 2[\frac{(1+(n+3)\epsilon+\delta)}{n+4} - \frac{(1+(n+3)\epsilon+\delta-\delta')}{n+4}]\\ = \;& \delta' - \frac{\delta'}{2} - \frac{2\delta'}{n+4} = \delta'(\frac{1}{2} - \frac{2}{n+4})\end{aligned}$$

and is strictly positive. This contradicts the optimality of σ.

In each of these five cases, we obtain a contradiction to the assumption that a new customer has been bribed. We may therefore assume that, without loss of generality, only old customers are bribed by the bribing strategy σ and there is at most one old customer that has not been fully bribed.

Finally, suppose that there is in fact one bribed old customer that has not been fully bribed. Let us call this old customer c and further suppose that it has been bribed δ where $0 < \delta < 1$. We will again show that this yields yet another contradiction. We have the capacity to increase this bribe to 1 at a cost of $1 - \delta$ (which we can do, given the remaining resource). The $\mathbb{P}$-ratings of all the customers within $N(c)$ will increase with the cumulative increase (only due to new pendant neighbours) being

$$n\frac{1+\epsilon}{2} - n\frac{\delta+\epsilon}{2} = n\frac{1-\delta}{2}.$$

Hence we obtain an increase in revenue for n sufficiently large ($n \geq 3$). This contradicts the optimality of σ. Henceforth, we assume that, without loss of generality, any optimal bribing strategy σ on (C, E), with initial evaluation $eval_0$, is necessarily such that only old customers are bribed and bribed old customers are fully bribed.

Suppose that the bribing strategy σ bribes less than k old customers; so, there is an old customer c that has not been bribed. We will show that this yields a contradiction. Let us amend σ to obtain a new bribing strategy σ' by bribing c so that $\sigma(c) = 1$. This costs us 1 unit of resource. There are no customers of C such that its $\mathbb{P}$-rating decreases, and the cumulative increase in $\mathbb{P}$-rating of the n new pendant customers adjacent to c is

$$n\left(\frac{1+\epsilon}{2} - \epsilon\right) = n\left(\frac{1-\epsilon}{2}\right) > \frac{n(2n+3)}{2(2n+5)} > \frac{n}{4}$$

which is strictly greater than 1 (the amount invested) for n sufficiently large ($n \geq 5$). This contradicts the optimality of σ. Furthermore, it is clear that more than k old customers could not have been bribed since the initial utility of the network totals only k and each old customer is bribed by 1. Therefore, we may assume that any optimal bribing strategy σ on (C, E), with initial evaluation $eval_0$, is necessarily such that only old customers are bribed, all bribed old customers are fully bribed, and exactly k old customers are bribed.

Consider the revenue accruing from our optimal bribing strategy σ. Irrespective of which k old customers are fully-bribed, the $\mathbb{P}$-rating of each of these old customers increases by

$$\frac{(1+(n+3)\epsilon)}{n+4} - \epsilon = \frac{1-\epsilon}{n+4}$$

and the $\mathbb{P}$-rating of the new pendant customers adjacent to each of these bribed old customers increases by

$$\frac{1+\epsilon}{2} - \epsilon = \frac{1-\epsilon}{2}.$$

All that remains is to compute the revenue accruing due to the new edge customers adjacent to each of these bribed old customers (as the $\mathbb{P}$-rating of any other old or new customer does not change). However, this will depend upon how many bribed old customers each new edge customer is adjacent to.

Let m_i denote the number of new edge customers adjacent to i bribed old customers, for $i = 1, 2$. If a new edge customer c is adjacent to 1 bribed old customer then its increase in $\mathbb{P}$-rating is

$$\frac{(1+\epsilon)}{2} - \epsilon = \frac{(1-\epsilon)}{2}$$

and if it is adjacent to two bribed old customers then its increase in $\mathbb{P}$-rating is

$$\frac{(2+\epsilon)}{3} - \epsilon = \frac{2(1-\epsilon)}{3}.$$

So, the total increase in revenue is

$$m_1\frac{(1-\epsilon)}{2} + m_2\frac{2(1-\epsilon)}{3}.$$

We also know that by counting the edges joining bribed old customers and their adjacent new edge customers, we obtain that $3k = 2m_2 + m_1$. Hence, the total increase in $\mathbb{P}$-rating due to new edge customers is equal to

$$m_1\frac{(1-\epsilon)}{2} + m_2\frac{2(1-\epsilon)}{3} = (3k-2m_2)\frac{(1-\epsilon)}{2} + m_2\frac{2(1-\epsilon)}{3} = \frac{3k(1-\epsilon)}{2} - m_2\frac{(1-\epsilon)}{3}.$$

So, the total increase in revenue due to the optimal bribing strategy σ is

$$\begin{aligned} & \frac{k(1-\epsilon)}{n+4} + \frac{nk(1-\epsilon)}{2} + \frac{3k(1-\epsilon)}{2} - m_2\frac{(1-\epsilon)}{3} - k \\ = \; & (1-\epsilon)[\frac{k}{n+4} + \frac{k(n+3)}{2} - \frac{m_2}{3}] - k. \end{aligned}$$

Clearly this revenue is largest when m_2 is 0, and if $m_2 > 0$ then the revenue is less than this maximal value. Also, when m_2 is 0 this revenue is exactly equal to ρ. Hence, as we started with a yes-instance of NVKL, we must have that no new edge customer is adjacent to two bribed old customers; that is, the k vertices of G corresponding to the k bribed old customers in C form an independent set, and (G, k) is a yes-instance of ISREG(3).

5.4.3 Summary of Reduction

We were initially given an arbitrary instance (G, k) of ISREG(3), and we constructed an instance $((C, E), eval_0, \rho)$ of NVKL. We have shown two things:

1. If (G, k) is a yes-instance of ISREG(3); that is, G has an independent set of size k, then there exists a bribing strategy σ on $((C, E), eval)$ that yields a revenue of ρ. This strategy is to bribe by 1 each of the k customers corresponding to the vertices of the independent set.

   In other words, (G, k) is a yes-instance of ISREG(3) implies that $((C, E), eval_0, \rho)$ is a yes-instance of NVKL.

2. If $((C, E), eval_0, \rho)$ is a yes-instance of NVKL; that is, there exists a bribing strategy σ such that σ yields a revenue of at least ρ, then it must be the case that σ can be manipulated into some other bribing strategy that bribes k customers 1 each and these customers correspond to vertices of G that form an independent set (of size k).

   In other words, $((C, E), eval_0, \rho)$ is a yes-instance of NVKL implies that (G, k) is a yes-instance of ISREG(3).

Theorem 5.1. *NVKL is **NP**-complete.*

Proof. This is a direct consequence of the above. □

5.5 Summary of NVKL

As we have seen, for the case of NVKL, the computation of weakly-dominant bribing strategies is non-trivial and requires a deep and thorough analysis of both the algorithmic and complexity-theoretic properties of the problem. Unfortunately, unlike the case of AVKL, we cannot simply follow a $\mathbb{P}$-greedy approach as we might have liked. In fact, even after addressing the apparent complications of the 'non-static' influence weight, we have seen that the greedy algorithm fails to find a weakly-dominant bribing strategy in general.

Perhaps surprisingly, we have been able to prove the **NP**-completeness of NVKL by showing how we can manipulate some arbitrary bribing strategy and give a polynomial-time reduction from the seemingly unrelated problem of ISREG(3). This is a very important result that can go a long way to addressing the issue of the 'bribery-proofness' of the rating-system. We can clearly construct instances of NVKL for which there exist weakly-dominant bribing strategies and we can provide specific examples of this, as is demonstrated in [2]. Although this may be the case, we can be suitably confident that these cannot be easily computed; that is, of course, unless $\mathbf{P} = \mathbf{NP}$.

The question from this point on is whether the case of NVKL is still salvageable from the perspective of the restaurant owner. Perhaps they can devise some way to compute an approximate or satisfactory solution that yields at least a positive return. More specifically, we may still be able to salvage something of the $\mathbb{P}$-greedy approach and Algorithm 2. We saw through our counterexample that whilst not yielding the optimal amount of revenue, we can still compute a profitable return. We approach, in the coming chapter, this question from a slightly less formal but nevertheless important angle and seek to obtain a number of experimental results concerning the performance of Algorithm 2 and others.

Chapter 6

Coping with NP-completeness

It would be reasonable to assume that proving the **NP**-completeness of a problem is somewhat terminal. After all, if $\mathbf{P} \neq \mathbf{NP}$ [20], then the best algorithms we could hope for would require super-polynomial time complexity in terms of the input size. In fact, there is a wide range of approaches we can take towards finding useful solutions to problems which are known to be **NP**-complete. Of course, **NP**-completeness does not just go away, but we can develop techniques to compute *good* solutions that are computationally feasible.

We can develop algorithms to compute solutions to a subset of instances of the original problem. For example, we know that the INDSET (independent set) problem is **NP**-complete [5], but if restrict our graph instances to trees then we can solve the problem in linear time using dynamic programming. Alternatively, we can develop approximation algorithms which guarantee us a solution optimal up to some fixed error ϵ, or randomised algorithms that guarantee us an optimal solution within some bounded probabilistic error.

Within this section, we will investigate two such methods by which we might *cope* with the **NP**-completeness of NVKL. We will develop a number of heuristic algorithms for solving the problem of computing bribing strategies under NVKL and provide an investigation into the relative quality of their solutions. In addition to this, we will analyse, on a more refined level, the complexity of algorithms for computing weakly-dominant bribing strategies under NVKL by showing that the problem is fixed-parameter tractable under practically realistic parameterizations of the NVKL problem.

6.1 Upper Bounds on Bribing Strategies

Having established that the problem of computing a weakly-dominant strategy in the case of NVKL is computationally intractable, we turn our attention towards computing profitable bribing strategies. There are several important questions we would be interested in investigating concerning the upper and lower bounds on returns from bribing strategies.

Aside from providing a definite limit on the profitability for any bribing strategy upon any network of customers, the motivation for finding such bounds is that they would provide a way for us to evaluate the quality of any approximating or heuristic algorithms that might be developed in future work. For example, an upper bound on the revenue of any bribing strategy would provide a useful way to score the effectiveness of a heuristic algorithm yielding *good* bribing strategies; that is, it would tell us how good the return is as, say, a percentage of the maximal possible return.

Proposition 6.1. *Let (C, E) be a customer network and let the set of voters V be such that $V \subseteq C$. Let the initial evaluation on (C, E) be $eval_0$ and let $\sigma \in \Sigma$ be an arbitrary efficient bribing strategy. Then*

$$r_{\mathbb{P}}(\sigma) < n$$

where $|C| = n$ and $\rho = u^0_{\mathbb{P}}$.

Proof. Recall that $r_{\mathbb{P}} = u^\sigma_{\mathbb{P}} - u^0_{\mathbb{P}}$. Let $u_{\mathbb{P}}$ be the utility of any arbitrary customer network (C, E) with some evaluation *eval*

$$u_{\mathbb{P}} = \sum_{c \in C} \mathbb{P}\text{-rating}(c, eval).$$

Since $|C| = n$ and $0 \leq \mathbb{P}$-rating ≤ 1, it follows that $u_{\mathbb{P}} \leq n$. As a result, we know that $u^\sigma_{\mathbb{P}} < n$. Since $u^0_{\mathbb{P}} \geq 0$ the result follows. □

It is not too difficult to come up with examples of networks and evaluations for which we can define bribing strategies which yield revenues close to this upper bound. Let us consider the following example.

Example 6.1. *Consider the following customer network which comprises two disjoint sub-networks. One sub-network (of size $n - 1$) consists of a star-graph of non-voters attached to a voter with an evaluation of ϵ. The other sub-network consists of a solitary voter of evaluation 1.*

The initial utility is given as $u^0_{\mathbb{P}} = 1 + (n-1)\epsilon$ and so this is the amount we can re-invest back into bribing the non-saturated customers of the network. If we take the strategy of bribing the voter at the centre of the star fully (by amount $1-\epsilon$) then we get that the utility after the execution of the strategy $u^{\sigma}_{\mathbb{P}} = (n-1) + 1 - (1-\epsilon) = n - (1-\epsilon)$. For ϵ taken to be arbitrarily small, we see that there exist examples for which a profit of $n-1$ can be obtained and the above bound is tight enough for practical use in evaluating the quality of heuristic algorithms.

6.2 Heuristic Approaches to NVKL

Heuristic methods attempt to yield good, but not necessarily optimal, solutions where it is infeasible to compute an optimal one. Heuristics are often based on *intuition* or a *rule of thumb* and have long since been employed for finding appropriate solutions to a wide range of important problems. It is widely agreed upon that good heuristic procedures should adhere to the following properties: a solution should be able to be obtained with reasonable computational effort; a solution should be near optimal (perhaps with high probability); and the likelihood of obtaining a solution that is far from optimal should be low [12]. We will see, for the problem of NVKL, that there is plenty of intuition we can use to motivate heuristic approaches towards computing bribing strategies.

6.2.1 Greedy Algorithm

In Chapter 5 we see that the decision problem NVKL is **NP**-complete. This leads us to begin thinking about whether we can compute good but not necessarily optimal bribing strategies. In order to do this we turn towards heuristic algorithms. Prior to showing that NVKL is **NP**-complete, we derived expressions for the projected revenue of bribing a (voting or non-voting) customer some positive and efficient amount. Furthermore, having established that it is actually not necessary to define an ordering on the execution of strategies in the presence of non-voters, this provided motivation to define a greedy algorithm for the computation of weakly-dominant bribing strategies. It is this algorithm to which we return. It has been established that clearly this greedy algorithm is not optimal, but how well (or poorly) it actually performs we do not know. It could enable us to compute reasonable bribing strategies.

6.2.2 Independent Set Heuristic

In order to prove that NVKL is **NP**-complete, we reduced the problem of ISREG(3) to NVKL. From an arbitrary instance of ISREG(3), we constructed the customer network by adding new vertices and defining an evaluation. From this instance we saw that the optimal bribing strategy was to bribe the old customers of the network which formed an independent set. As a result we can deduce that there are instances, not too far from an arbitrary graph instance, for which the optimal bribing strategy is to greedily bribe the customers of an independent set. It therefore begs the question as to whether the heuristic of looking for and bribing independent sets in customer networks produces good results.

However, INDSET is also **NP**-complete [5]. Although this is the case, it is a fundamental problem in algorithmic graph theory and and there exists a number of heuristic approaches towards finding approximate independent sets in polynomial-time. For example, several approaches can be found in [21].

In order to construct an approximate independent set we use the following simple algorithm from [21]. By a theorem from [24], Algorithm 3 outputs an independent set S such that $|S| \geq n/(\Delta + 1)$ where Δ is the maximum degree of any vertex of G.

```
Data: Graph $G(V, E)$.
Result: Approximate independent set $S$.
$S = \emptyset$;
while $V \neq \emptyset$ do
    Let $v$ be a vertex of minimum degree in $V$;
    $S = S \cup \{v\}$;
    Remove $v$ and its adjacent vertices from $G$;
end
return $S$
```

Algorithm 3: A greedy algorithm for approximate independent sets.

We use this algorithm for computing approximate independent sets for our independent set heuristic algorithm for the computation of a bribing strategy. We give this algorithm as follows (note that we use the earlier results of Propositions 5.1 and 5.2 to determine whether a customer is profitable to bribe).

```
**Data:** Customer network $(C, E)$ and initial evaluation $eval_0$.
**Result:** Bribing strategy $\sigma$.
$\sigma(c) = 0$ for all $c \in C$;
$budget = u^0_{\mathbb{P}}$;
while $C \neq \emptyset$ do
    Let $c$ be a customer of minimum degree in $C$;
    if $c$ is profitable to bribe then
        $\sigma(c) = min\{1 - eval(c), budget\}$;
        $budget = budget - \sigma(c)$;
    end
    Remove $c$ and its adjacent customers from $(C, E)$;
end
return $\sigma$
```

Algorithm 4: Independent set heuristic for bribing strategies.

6.3 Computational Evaluation

The ideas discussed above give us an intuition as to what a sensible way to bribe customers might be. From these ideas we have developed algorithms that we expect might compute reasonable bribing strategies where it is not feasible to compute optimal ones. In order to obtain an empirical evaluation of their performance, we have implemented the algorithms (as well as the optimal algorithm in the case of AVKL). Within this section, we will provide a computational evaluation of the results obtained by each algorithm on a range of different networks. These networks are constructed from random graph models for social networks (see Appendix A). By performing these experiments, we hope that we will be able to demonstrate good performance of heuristic approaches to computing profitable (yet non-optimal) bribing strategies for the **NP**-complete problem of NVKL and provide a comparison of the various approaches we have developed. Furthermore, we will comment on the properties of the customer networks we bribe, as we will see that these closely affect our results.

6.3.1 Weakly-Dominant Strategies under AVKL

Firstly, we implement, in the case of AVKL, an algorithm for computing a weakly-dominant bribing strategy. This might give us an indication, for certain classes of networks (or models by which we generate networks) as to the revenue levels a weakly-dominant strategy can yield. Two important network parameters to investigate are the size of the network (the number of customers) and the average degree of the network (this tells us, for an arbitrary customer, the average number of neighbours it has; that is, customers that affect its $\mathbb{P}$-rating). The two random graph models for social networks, Watts & Strogatz and Barabási-Albert, generate small-world and scale-free networks, respectively (see Appendix A).

The following plot shows the average revenue generated as a percentage of the initial utility of the network, against the size of the network. For each network size (ranging from 10 $\rightarrow$ 250 customers with a step size of 10), we take the average revenue (as a percentage of the initial utility) over 1000 independent runs. We do this for networks generated by both the Watts & Strogatz and Barabási-Albert models.

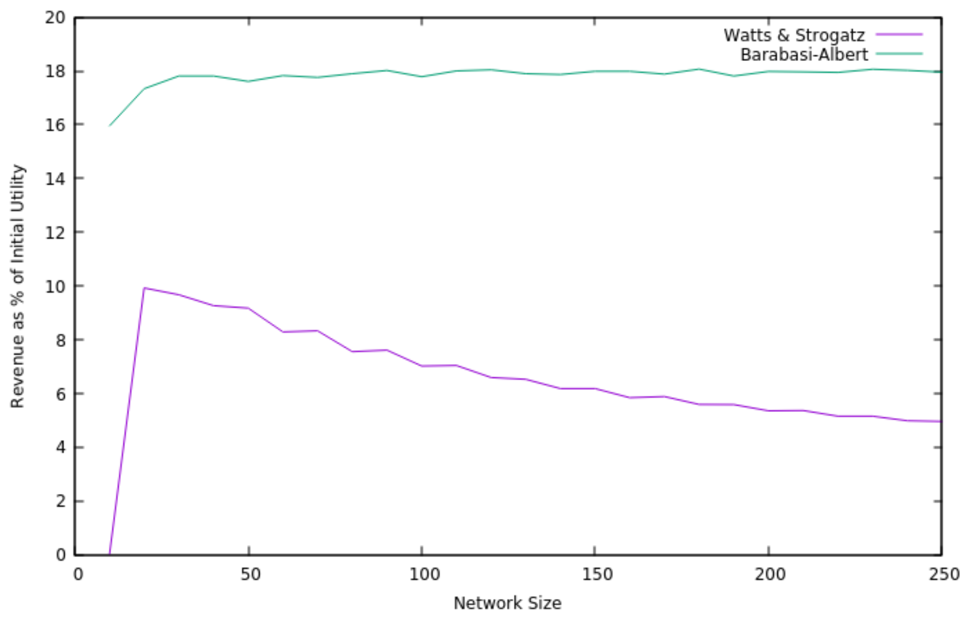

Figure 6.1: Network size vs. profit for weakly-dominant bribing strategies under AVKL.

As can be seen in Figure 6.1, weakly-dominant strategies consistently yield around an 18% profit (of initial utility) for Barabási-Albert networks; furthermore, this seems to be independent of the size of the network. Conversely, for Watts & Strogatz networks, the average profit is much lower and decreases as the network size grows (although this seems to settle around a profit of 5.5% when the network size is big enough).

Scale-free networks (as generated by the Barabási-Albert model) result in *hubs* around certain influential individuals. As a result, we would expect there to be customers that have very high influence when bribed. This is one potential explanation of the fact that the results show higher revenues for Barabási-Albert networks; that is, the greedy algorithm will bribe customers at the centres of these hubs and doing this should lead to higher revenue gains.

Using the Watts & Strogatz model, we can closely control the desired average degree of vertices of the network during construction. In customer networks, this corresponds to the average number of neighbours that customers have. An interesting question is to investigate is whether (and if it does, by how much) the average degree of customer networks affects the revenue generated by weakly-dominant bribing strategies. Figure 6.2 shows the average revenue obtained (over 1000 independent runs) for customer networks of size 100 with average degrees running from 5% to 100% (of the network size) with a step size of 5%. Clearly the average degree of the customer networks closely affects

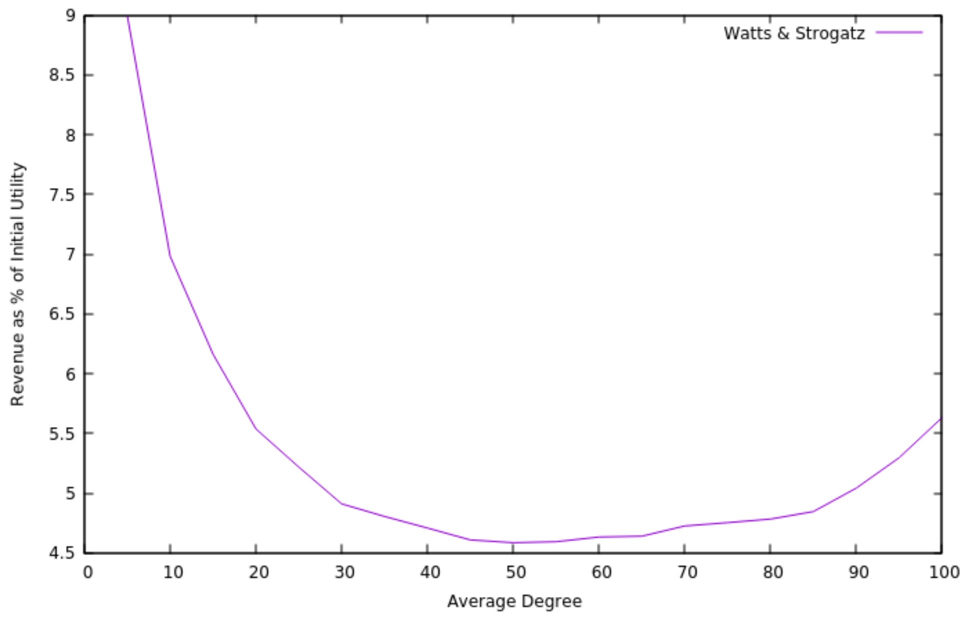

Figure 6.2: Average degree vs. revenue gained with a weakly-dominant bribing strategy.

the revenue generated by weakly-dominant bribing strategies. For small average degrees (customer networks where there is little influence between customers in relation to the network size) we see much higher levels of revenue generated. This may be due to the fact that in small neighbourhoods, single customers make bigger contributions towards the $\mathbb{P}$-ratings of their neighbours. The revenue starts to pick up slightly as the graph network approaches a clique-like structure.

6.3.2 Heuristic Approaches under NVKL

We are interested in gauging the performance of heuristic algorithms in terms of the revenue that they generate. In order to standardise this across all network sizes we can measure this as a percentage of the initial utility of the network, which gives a clear performance measure. As is discussed in Appendix A, there is a wide variety of models we can use for generating networks of voters. Different approaches enable us to control more closely different parameters of the network during construction.

Furthermore, of the proposed heuristic approaches (the greedy heuristic and the independent set heuristic), it would be useful to understand how well they perform both in absolute terms and relative to each other. In addition, we should attempt to analyse the network properties that affect the performance of both algorithms.

Firstly we generate networks of both voters and non-voters using a Watts & Strogatz model on 100 customers with an average degree of 10. An obvious point of interest is to investigate how the proportion of non-voters in a network affects the amount of revenue gained by the strategies computed by both algorithms. We can construct networks with an approximate proportion of non-voters. We begin with an AVKL network; that is, we begin with a proportion of 0%. We then increase this in a step size of 5% all the way up to 95%. For each proportion, we compute the average revenue gained over 1000 runs of both algorithms. The results can be seen in Figure 6.3.

We see that the greedy heuristic consistently outperforms the independent set heuristic for all non-voter proportions. Although this is the case, both heuristics give profitable bribing strategies which increase in revenue as the proportion of non-voters in the network grows. An important point to note is the performance of the greedy algorithm at a proportion of 0% non-voters. In this case, we have an all voter network and the problem we are trying to solve is that of AVKL. For networks consisting entirely of voters, a point that we discussed heavily in Chapter 5 is that influence weights are static. In fact, for networks of all voters, the greedy algorithm *is* the polynomial-time algorithm of [2] and yields a weakly-dominant strategy. We can see that this is clearly the case if we note that for voters, the projected revenue gained when bribing a voter which we showed in Proposition 5.2 is identical to the influence weight of AVKL (Definition 4.1). For an

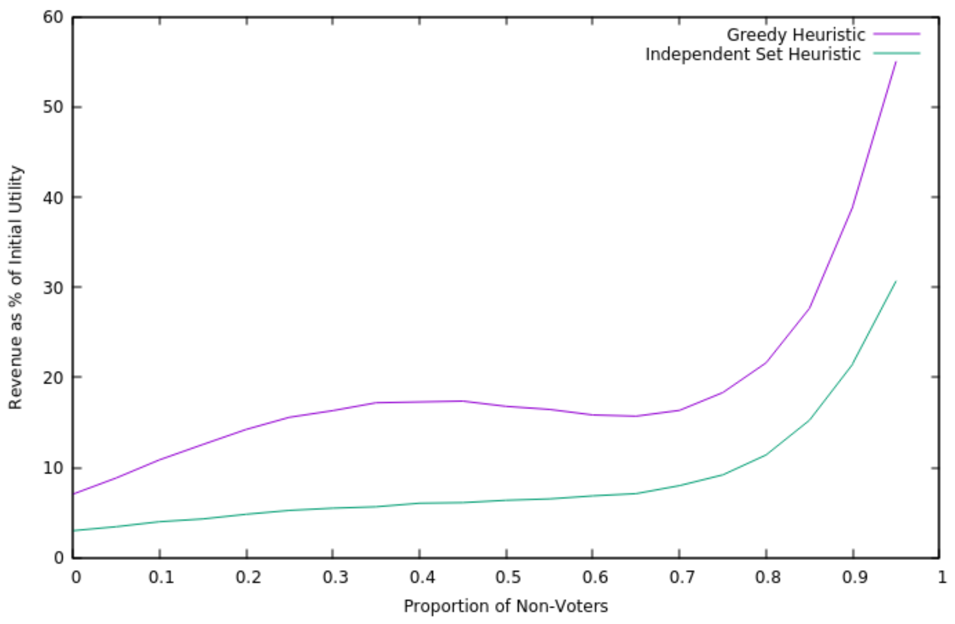

Figure 6.3: Proportion of non-voters vs. revenue gained for two heuristic approaches to NVKL.

all-voter Watts & Strogatz network on 100 customers with an average degree of 10, the algorithm of [2] yields a revenue of approximately 7.0204%. The greedy algorithm on an all-voter network yields a revenue of 7.0424% and so the results show that the cases of AVKL and NVKL align in experimentation as we would expect them to in theory.

The results obtained for the heuristic approaches to NVKL are generally quite good. At 0% non-voters we gain the same revenue as a weakly-dominant strategy does in AVKL. As the proportion of non-voters is increased we see a steady rise in revenue from both heuristic approaches. For high proportions of non-voters we are even able to obtain in revenue around 30% of the initial network utility.

As was done in the case of AVKL, another natural question to pose is how the revenue generated by either approach is affected by the size of the customer network. In order to investigate this question, we can generate a customer network using the Watts & Strogatz model with an average degree of 10% of the network size (we choose this number somewhat arbitrarily, but previous results at 10% yield good revenues). We set the proportion of non-voters as 30% for the same reason. Finally, we measure the average revenue gained from the bribing strategy produced by each heuristic over 1000 independent runs with network sizes ranging from 10 to 250 with a step size of 10. The results are can be seen in Figure 6.4. The trends observed seem to mirror those of AVKL and Watts & Strogatz models. For network sizes sufficiently large, the revenue settles to a steady point

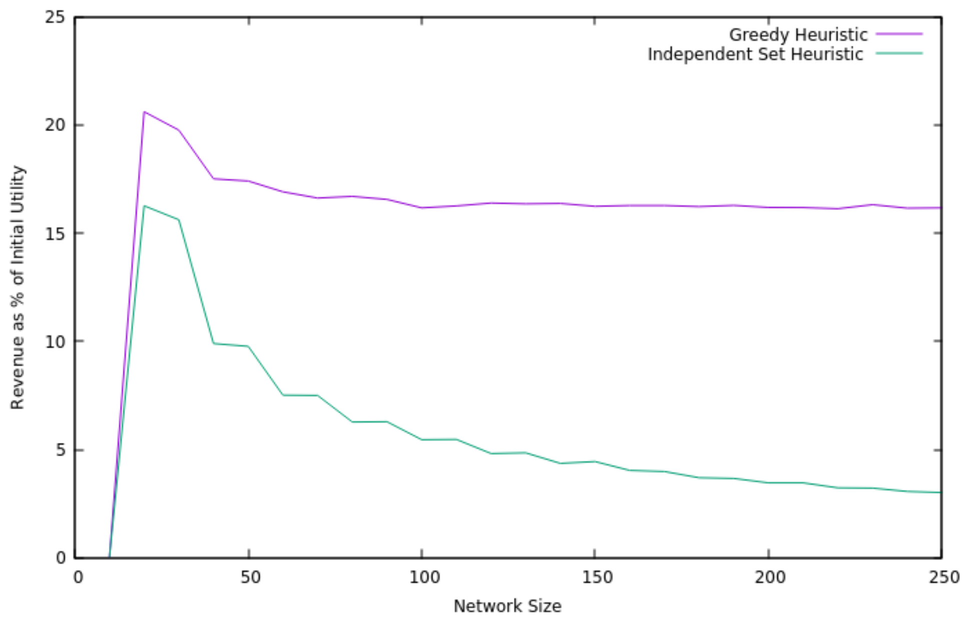

Figure 6.4: Network Size vs. revenue gained for two heuristic approaches to NVKL (Watts & Strogatz).

as the size increases. This is true of both heuristics, although the greedy algorithm, again, outperforms the independent set heuristic (although both consistently compute profitable bribing strategies).

We also conduct a similar experiment using networks constructed by the Barabási-Albert model. We use networks within the same size range and average results over 1000 independent runs. The results are shown in Figure 6.5 which indicates similar trends and relative performance between the two algorithms. Although this is the case, the revenue seems to reach a stable point more quickly in the case of Barabási-Albert and this stable point is a slightly lower revenue value than on Watts & Strogatz networks.

Finally, we investigate the relationship between the average degree of vertices of the random graph (or the average number of neighbours per customer) and the revenue that either algorithm yields. In order to investigate this property, we generate a Watts & Strogatz random graph of size 100 and compute the average revenue returned as a proportion of the initial utility of the network over 1000 independent runs. We begin with an average degree of 10 and move towards an almost clique structure with an approximate average degree of 100 (note that in a 100-clique all vertices would have a degree of exactly 100). We do this with a step size of 10. Figure 6.6 shows the results for both the greedy algorithm and the independent set heuristic.

We see that for the greedy algorithm, the revenue is largely stable (at around 17% of

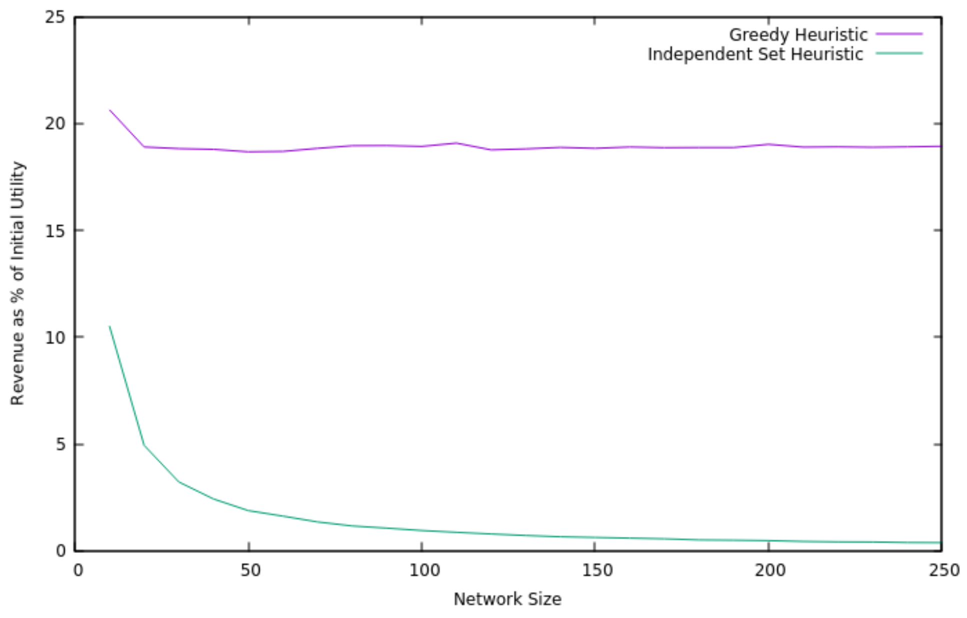

Figure 6.5: Network Size vs. revenue gained for two heuristic approaches to NVKL (Barabási-Albert).

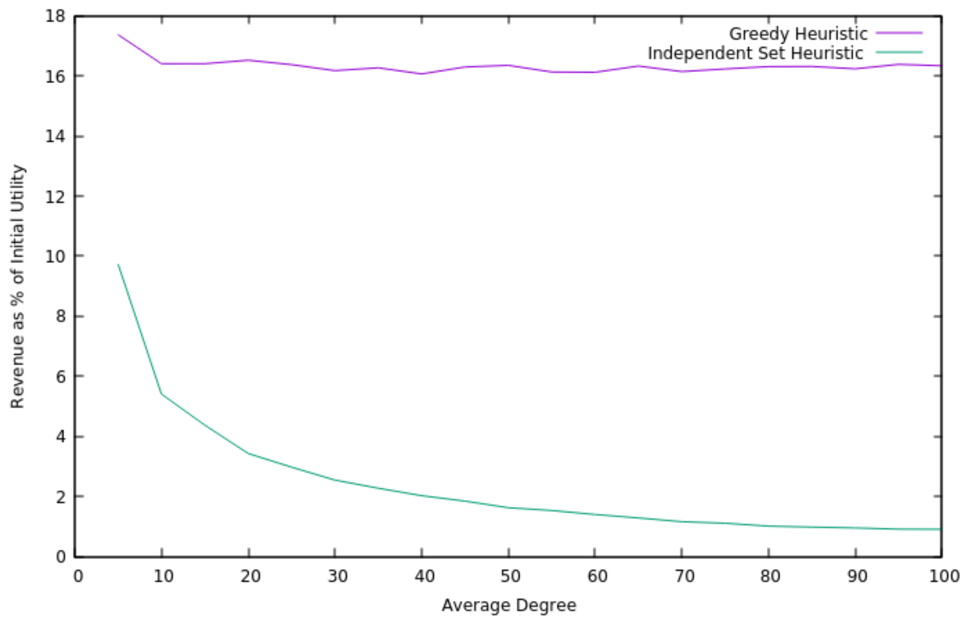

Figure 6.6: Average degree vs. revenue gained for two heuristic approaches of NVKL.

initial utility) in relation to the average degree. In the case of the independent set heuristic, we see a sharp decline in revenue (from around 10%) as the average degree increases. Eventually, the revenue seems to settle around 1.5% for higher average degree values. In a clique graph, by definition, there is no independent set (recall, every pair of vertices are incident). The aim of the independent set heuristic is to bribe an independent set of customers with the intuition being that this should yield a reasonable revenue. Clearly as the graph approaches a clique-like structure (as the average degree increases) we are less likely to find an independent set (or anything 'close'). Conversely, for smaller average degrees it is far more likely that we find independent or nearly independent sets of reasonable size which might account for the increased revenue.

6.3.3 Network Properties Affecting Performance

Having carried out an extensive computational evaluation of the algorithms for computing both weakly-dominant and approximate bribing strategies of AVKL and NVKL customer networks, we have seen that there are clearly a number of factors that affect the quality of our solutions. These factors include basic network parameters such as size and average degree, and also properties unique to customer networks such as the proportion of non-voters within the network (in the case of NVKL). Under certain values of these parameters, we see strong performance of the heuristic algorithms for NVKL; under others, we see weaker performance. As such, we would like to be sure that these algorithms perform well for values of these networks parameters that we might reasonably expect to see in real social or voter networks. Given these experimental results, there is a strong case to be made that this is so.

The results produced by both heuristic algorithms are generally quite good. We are able to consistently compute profitable bribing strategies for all ranges of parameter values. Furthermore, we are able to do so with reasonably little computational effort (polynomial-time: $\mathcal{O}(n^3)$ for the greedy heuristic and $\mathcal{O}(n^2)$ for the independent set heuristic). As we have discussed in Appendix A, the models for generating random graphs that we have used exhibit many of the characteristics that manifest themselves in real networks of voters. As such, we can be confident that these algorithms would perform well in practice as they have done in this computational experimentation.

6.4 Summary of Findings

Having proven the **NP**-completeness of NVKL, we have been able to develop solutions to tackle instances of the NVKL problem that we might expect to encounter in actual voter networks. These algorithms perform reasonably well and we have discussed the interesting intuition behind them and why this might be the case.

Moving on from the problem of NVKL, we look at the final case that was introduced in [2]. This is the case where we have non-voters and the locations of customers are not known to the restaurant owner.

Chapter 7

Non-voters unknown networks (NVUL)

In this section we consider the case where there are non-voters and the restaurant owner does not know the locations of customers on the network. More formally, we say that the restaurant knows the network of influence between customers, that is, $E \subseteq C \times C$, where here C is the set of (unnamed) customers. So, we can think of our network as a graph where there are no customer names on the vertices. Denote the set of customer names by N. The restaurant knows the evaluation function $eval : N \to Val \cup \{*\}$; that is, the restaurant can associate an evaluation to every customer name. However, what is missing is that the restaurant does not know where each customer name sits within the network; that is, the restaurant is missing a bijection between N and C. Of course, the restaurant knows the names of the customers who have voted and those who have not but the restaurant does not know the locations of the named customers within the network. In particular, the restaurant does not know the initial evaluation (as in order to compute this, the restaurant would need to know the locations of the customer names).

7.1 Expected Revenue of Bribing Strategies

Given that the restaurant does not know which customer each evaluation corresponds to, there is no way for the evaluation or bribe of some individual customer to be evaluated. Therefore it only makes sense for us to talk about the revenue generated by some bribing strategy in expectation. That is, we must consider the $n!$ different bijections between N

and C. Given some bribing strategy σ we define the expectation of the revenue of σ over (C, E) as the average over all possible permutations of customers.

$$\mathbb{E}[r_{\mathbb{P}}(\sigma)] = \sum_{\rho} \frac{1}{n!}[u_{\rho}^{\sigma} - u_{\rho}^{0}]$$

where u_{ρ}^{σ} denotes the utility of the network after executing σ under some permutation of the customers of the network ρ. Similarly u_{ρ}^{0} denotes the initial utility of the network under the same permutation.

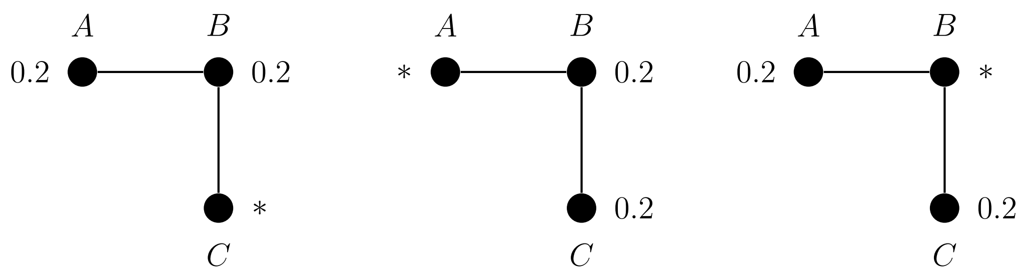

Figure 7.1: Permutations of the evaluation assignment up to symmetry.

Under NVUL, as is shown in [2], it is in fact possible to define bribing strategies that are profitable in expected return. An example taken from [2] is as follows. Consider the following network consisting of three customers A, B, and C. Let $eval(A) = eval(B) = 0.2$ and let C be a non-voter with $eval(C) = *$. The network is depicted in Figure 7.1.

Let $\sigma : C \to Val$ be the strategy of always bribing the non-voter by 0.2. That is, up to symmetry, for the three pictures permutations of evaluations, we have three bribing strategies σ_1, σ_2, and σ_3.

$$\sigma_1(A) = \sigma_1(B) = 0, \sigma_1(C) = 0.2 \implies r_{\mathbb{P}}(\sigma_1) = 0.2$$

$$\sigma_2(B) = \sigma_2(C) = 0, \sigma_2(A) = 0.2 \implies r_{\mathbb{P}}(\sigma_1) = 0$$

$$\sigma_3(A) = \sigma_3(C) = 0, \sigma_3(B) = 0.2 \implies r_{\mathbb{P}}(\sigma_1) = 0.1$$

Thus, σ_i is profitable for all 3! permutations of assignments of evaluations to customers (the remaining three cases are covered by symmetry). Therefore, we are able to define bribing strategies that are profitable in expectation under NVUL.

7.2 Utility Maximising Placement

Moving away from expectation of revenues under non-voters and unknown networks, there are additional problems that a restaurant owner might be interested in solving. It would be reasonable to think of this case as a generalisation of NVKL and as such, we would not expect that we should be able to compute weakly-dominant bribing strategies given the fact that this problem is **NP**-hard under NVKL. As we will see, it turns out that under the conditions of NVUL, it is not at all easy for the restaurant owner to deduce anything of particular interest about the utility of the restaurant nor to gain any insight into what a sensible bribing strategy might be.

The restaurant owner may or may not know (anything about) the overall utility of his or her restaurant in addition to the customer network and evaluations that he or she already separately knows. Supposing the restaurant owner knows the overall utility, he might hope to be able to compute in polynomial-time a feasible assignment of evaluations to customers such that the resulting utility of this matching is equal to the one the restaurant owner knows about. This would leave the restaurant owner in the case of NVKL within which we know that we can compute a profitable bribing strategy in polynomial-time. Perhaps the restaurant owner does not know what the utility of the restaurant is. In this case they might wish to be able to infer some kind of upper or lower bound or to discover whether it could possibly be that their utility surpasses a certain value.

Perhaps the most interesting of these problems, is the problem of assigning evaluations to customers such that the resulting utility of the customer network is maximised. Formally, we define this as follows.

Definition 7.1. *(UPLACE) Given a customer network C, an evaluation vector $eval \in \mathbb{Q}^m$ such that $m \leq |C|$, and a target utility $b \in \mathbb{Q}$. A yes-instance of UPLACE is such that there is an assignment of evaluations to customers such that: (a) every customer is adjacent to at least one voting customer and (b) the utility resulting from the assignment is at least b.*

It turns out that we can show that UPLACE is **NP**-complete by a simple reduction from the vertex dominating set problem.

Definition 7.2. *(VDS) Given any graph G and $k \in \mathbb{N}$ such that $k \leq n$ where n is the number of vertices of G. A yes-instance of VDS is an instance where there is a subset S of k vertices so that every vertex not in S is adjacent to at least one vertex of S.*

Claim 7.1. *UPLACE is **NP**-complete.*

Proof. Given any instance (G, k) of VDS we construct a customer network $C = G$. We define $eval = (1, ..., 1) \in \mathbb{R}^k$ and let $b = k$. Let $(C, eval, b)$ be an instance of UPLACE. If (G, k) has a vertex dominating set of size k then clearly the assignment of 1 to each customer corresponding to a vertex of the dominating set would give us a utility of $b = k$ in C. Conversely, if G has no vertex dominating set of size k then there is no assignment of evaluations 1's to customers such that condition (a) of Definition 7.1 holds. That is, $(C, eval, b)$ is not a yes-instance of UPLACE. Since VDS is **NP**-hard [19] it follows that UPLACE is **NP**-hard. Given an instance of UPLACE we can clearly check in non-determinstic polynomial time whether the utility of C with respect to $eval$ is $\geq b$. We can therefore deduce that UPLACE is in NP and the result follows. □

As we can see there is little scope for the restaurant owner to accomplish anything of particular interest under the existence of non-voters and given an unknown network.

7.3 Fixed-Parameter Tractability of UPLACE

Although we have shown that the decision problem of matching evaluations to customers so as to maximise the utility of the network is **NP**-complete, we could still expect to be able to solve the problem given instances that are likely to arise in practice. Under some natural parameterization of UPLACE, for example, the number of non-voters of the network (C, E), we may hope to be able to find solutions efficiently as long as the number of non-voters remains small. Unfortunately, as we will now show, UPLACE is **W**[**2**]-hard. We recall, in this proof, several key definitions from the background section on parameterized complexity theory (see Section 2.5).

Proposition 7.1. *UPLACE (where our parameter is the target evaluation) is **W[2]**-hard.*

Proof. Consider the reduction from VDS to UPLACE of 7.1. In addition to this, we note that VDS is **W**[**2**]-hard (where our parameter is the size of the vertex dominating set) by a theorem from [25]. We claim that this reduction is in fact an fpt-reduction. To show this, we must prove the following three properties of the reduction (which we will call R). Recall the definition of an fpt-reduction (see Definition 2.40).

1. Trivially, for some instance x of VDS, $x \in$ VDS if and only if $R(x) \in$ UPLACE.

2. R simply outputs the graph G along with an evaluation vector consisting of all 1's. The size of this graph is at most $\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$ which is $\mathcal{O}(n^2)$. The size of the evaluation vector is k. Therefore the reduction takes $\mathcal{O}(n^2) + \mathcal{O}(k)$ which is of the form $f(k) \cdot p(n)$ for some polynomial p.

3. Finally, we note that the parameterization of the input instance x is exactly equal to the parameterization of $R(x)$.

We conclude that R is an fpt-reduction. Since **W[2]** is closed under fpt-reductions (see Section 2.5), the result follows. □

7.4 Summary of NVUL

As we can see, other than to consider the expectation of bribing strategies under NVUL, there is perhaps not much of use that a restaurant owner could do. It does not make sense to talk of computing weakly-dominant strategies given that we do not know how to find a customer's evaluation. Furthermore, we see that it is hard even to compute the assignment of evaluations that would yield a given utility; and that even when we consider the parameterized case, this problem is **W[2]**-hard.

Chapter 8

Project Evaluation

The main purpose of the work of Grandi and Turrini [2] was to introduce the network-based rating system and provide an analysis of how some malicious agent might manipulate the opinions of the individuals of this network, through bribery, in order to increase his or her gain. The analysis provided was largely restricted to the case of AVKL and, in addition to this, did not explore in any way the complexity theoretic aspects of the problem of bribery. As we identified in the initial stages of the project, a main aim of this project was to extend the work of [2] beyond only these cases. Furthermore, it appeared as through the differing knowledge cases of the restaurant owner would complicate the computation of weakly-dominant bribing strategies and provide interesting problems to investigate in relation to their computational complexity. As we have seen, this has indeed been the case.

8.1 Summary of Accomplishments

We have further developed the framework presented in [2] by providing several original theoretical results. We have also proven important complexity-theoretic properties of the problem of bribery in multiple different knowledge cases, and proposed and implemented feasible alternative approaches towards solving the problem in practice.

We began by taking the problem of computing weakly-dominant bribing strategies in the case of AVKL. As our starting point, we identified that the existence of non-voters complicates matters, and that the approach taken by Algorithm 1 would no longer work due to the fact that influence weight is no longer static. We sought to devise a similar approach by investigating the revenue gained by bribing voters and non-voters under NVKL. We also sought to investigate how the order in which they are bribed affected the revenue gained. We proved:

1. The NVKL analogue of Theorem 4.1:
   (a) Proposition 5.1 gives us the projected revenue of bribing a non-voter.
   (b) Proposition 5.2 gives us the projected revenue of bribing a voter.
2. The order in which we bribe any two customers some pre-determined amount does *not* affect the revenue gained.
   (a) Proposition 5.3 shows we may bribe two non-voters in any order.
   (b) Proposition 5.4 shows we may bribe a voter and a non-voter in any order.

We used these results to develop a polynomial-time greedy algorithm (Algorithm 2) for computing bribing strategies under NVKL.

Unfortunately we saw that, in fact, this greedy algorithm is sub-optimal in certain cases. As such, we see that NVKL is markedly different from AVKL. A full proof of this fact was given in Example 5.1. From here, we turned our attention towards thinking about whether even computationally efficient algorithms for computing weakly-dominant bribing strategies under NVKL actually exist.

In Section 5.4, we showed that this could not possibly be the case by providing a detailed and complicated proof that the problem of computing weakly-dominant bribing strategies under NVKL is **NP**-complete. Of course, we are assuming that $\mathbf{P} \neq \mathbf{NP}$. We then proceeded to tackle the problem of computing approximate or satisfactory bribing strategies under NVKL.

We discussed the motivation for and developed two heuristic algorithms for bribing strategies under NVKL. These are the initial greedy Algorithm 2, and the independent set heuristic used by Algorithm 4. Additionally, we performed an experimental analysis of our approaches to solving NVKL. This involved implementing the framework presented in [2] and implementing both of our algorithms in Java. In addition to this, we introduced and implemented a number of random graph models for generating realistic social networks.

Having implemented our algorithms, we attempted to explain why the algorithms performed as they did under the graph and customer network parameters we experimented with in Section 6. By doing this, we were able to comment on the potential practical feasibility of our algorithms on real-world networks that exhibit the properties that we attempted to model through our random graph models. (See both Chapter 6 and Appendix A.)

Finally, we began to investigate the remaining case of NVUL (Chapter 7). We established that it does not make sense to talk about the problem of a weakly-dominant bribing strategy under NVUL and so we formulated the related problem of UPLACE (see Definition 7.1). In Claim 7.1, we showed that UPLACE is **NP**-complete. We concluded this chapter by showing that UPLACE is also **W[2]**-hard.

8.2 Evaluation

It is a central aim of the research conducted in [2] to determine the resistance (of the proposed rating system) to bribery by some malicious agent. The main conclusions of the paper are that in general the $\mathbb{P}$-rating is *not* bribery proof under each of the knowledge cases that are investigated. The extent to which it is established that the $\mathbb{P}$-rating is not bribery proof (under the case of NVKL) is simply by showing that there exist networks and bribing strategies that yield a positive revenue. An algorithmic approach to actually computing these strategies, in general, is not considered.

It is mentioned in [2] that it is hard to obtain analytical results for strategies bribing non-voters. The results we have shown concerning the bribing (and the order of doing so) of customers under NVKL (Propositions 5.1, 5.2, 5.4, 5.3) go a long way towards exploring this. We have shown the results required to attempt to formulate an algorithm that loosely follows the $\mathbb{P}$-greedy method of [2] that turns out to yield a weakly-dominant strategy for AVKL. This is despite the complicating matter of the stated 'non-linearity of the $\mathbb{P}$-rating' given the existence of non-voters. It is perhaps alluded to in [2] that a similar approach to NVKL would work as it is stated in Proposition 12 of [2] that for bribing strategies restricted to voters only, the influence weight of AVKL still applies. We have been able to show that this is, in fact, not the case by providing a counterexample to the greedy approach that we proposed in Algorithm 2.

It is also perhaps implicitly assumed that the problem of computing weakly-dominant bribing strategies under NVKL is actually tractable, as was the case under AVKL. The matter of tractability (or the computational complexity) of computing bribing strategies is not contemplated in the paper. In actual fact, when we are considering the resistance to bribery of the proposed rating system, it is an important point to consider. We see that it is possible to construct networks for which there exist profitable bribing strategies and it is concluded that the $\mathbb{P}$-rating is therefore not bribery proof. It could be the case that in general, given a network, it requires a considerable amount of computational effort to compute such strategies.

Having formally proven that the problem of computing weakly-dominant bribing strategies is **NP**-complete (see Section 5.4), this clearly leads us to rethink our definition of *bribery-proof*. Although weakly-dominant strategies might exist, we can be sure that, in general, these can not feasibly be computed. Ideas such as this have long since been explored in other frameworks and other areas of computational social choice. The case is put forward in [4], that computational complexity can play a role in "protecting the integrity of social choice" and it is remarked that many standard voting schemes can be manipulated with very little computational effort. We see through our own results, that for the rating-system of [2] there are significant complexity-theoretic barriers to manipulation, particularly under the cases of NVKL and NVKL.

Chapter 9

Conclusion

During this project, we have studied several interesting algorithmic problems resulting from the network-based rating system of [2]. We have seen how we can employ techniques from a variety of disciplines in mathematics and computer science to prove non-trivial and original results concerning both algorithmic and complexity-theoretic properties of the framework.

Although we have been able to show a number of important results, the rating-system of [2] and the problem of bribery provide much scope for much further investigation into the algorithmic and complexity-theoretic properties of the framework.

9.1 Future Work

There are a number of interesting problems that we initially identified to be interesting (see Chapter 3). Some of these we have exhaustively investigated and some of these still remain unexplored due to lack of time. In addition to the problems discussed in Chapter 3, there are a number of problems we encountered during our work, that would certainly be worth looking at and would help further refine our analysis of the rating system and the complexity of bribery within it. We will focus on the problems that are more closely related to algorithms and complexity theory. In no particular order, these include the following.

1. **Determining whether the problem of NVKL (see Definition 5.1) is fixed-parameter tractable.** Parameterized complexity theory enables us to provide

a further refined analysis of hard algorithmic problems. In earlier sections, we have discussed the network structures and properties that we might expect to see in real-world networks (see Appendix A). As a result, it would be reasonable to assume that there exist parameterizations of the **NP**-complete problem of NVKL so that we know that the parameter will always remain comparatively small (in relation to the input size). For example, such a parameter could be the number of non-voters k given an instance $((C, E), eval_0)$ of size n (where $k \ll n$). This is simply a natural parameterization at a first glance; in fact, parameterizations yielding fixed-parameter tractable solutions could be entirely non-trivial and a more focused analysis of the 'source' of the **NP**-completeness of NVKL would be required. Alternatively, we may not be able to find a parameterization of the problem under which it is fixed-parameter tractable. In this case we might aim to prove a hardness result; that is, to show that NVKL is **W**[**P**]-hard [22]. Having thought about this problem, we have not been able to prove a result and it seems that such a result would be extremely difficult to show. One approach could be to note that the independent set problem (INDSET) is a somewhat classic example of a **W**[**1**]-complete problem [22]. Although we reduce this problem to NVKL in showing that it is **NP**-complete, this is certainly not an fpt-reduction; we could begin by attempting to find one which is. In any case, a result on the fixed-parameter complexity of NVKL would enable us to determine with more certainty whether we are able to easily compute exact solutions to instances of the problem that are likely to arise in practice.

2. **Studying the hardness of approximation of NVKL**. The NVKL problem may also be viewed as an *approximation problem*. Since we know that NVKL is **NP**-hard, it would be interesting to ascertain whether we can develop algorithms which we can formally prove are guaranteed to return a solution within some bound of the provably optimal one. We have provided some brief insight into how we might bound the revenue generated by any bribing strategy given some network; however, to develop good approximation algorithms we would first look at tightening our bound (almost surely as a function of both the network and the initial evaluation and utility). We have developed heuristic algorithms which do perform well in our experimentation. This is all well and good, but as we have seen, the performance of our algorithms varies massively depending on the types of network and their parameters. In certain cases we struggle to generate very much revenue at all and can 'fail' to produce good results (although, as we have discussed, these instances are

not likely to arise in practice). The point here, is that we have no formal guarantee as to the performance of heuristic algorithms, and approximation algorithms would provide a more formal reassurance that we are able to efficiently compute good solutions in the general case.

3. **Generalisation of the $\mathbb{P}$-rating and its framework**. It would be interesting to see whether the results we have proven are extensible to several of the generalisations of the $\mathbb{P}$-rating that we proposed in the initial stages of the project. For a full description of these problems, see Chapter 3. In brief, these were:

   (a) $\mathbb{P}$-ratings on weighted networks of influence.

   (b) $\mathbb{P}$-ratings on wider spheres of influence (beyond a customer's immediate neighbourhood).

   (c) $\mathbb{P}$-ratings on wider classes of functions beyond the average of the neighbourhood (this could include the mode, median, or even arbitrary classes of functions that possess some property, e.g. additivity).

   We would like to answer the question of under which of these generalisations can we compute weakly-dominant bribing strategies in polynomial-time? In relation to the $\mathbb{P}$-rating and the above generalisations, it would also be interesting to determine the restrictions of customer networks (e.g. trees or star graphs) which allow us to solve hard problems such as NVKL. Answering questions such as these would increase the feasibility of the $\mathbb{P}$-rating for implementation as an actual rating-system.

   Finally, we saw that our **NP**-completeness result holds when customer evaluations can be rational numbers from $[0, 1]$. In the proof of the polynomial-time reduction from ISREG(3) to NVKL, this allowed us to choose ϵ so that our initial utility was exactly k (the size of the independent set we were looking for in the instance of ISREG(3)). If Val were, e.g., $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$ (which would correspond to the 'traditional' five-star rating system) then we would need to control the initial utility in some other way. It would be interesting to see whether the actual set *Val* affects the computational complexity of NVKL and, if so, determine exactly the form of this effect.

Appendix A

Implementation of Framework

Whilst the rest of this project concerns the tractability of bribery problems resulting from the various restrictions we can place upon customer networks, we are still interested in evaluating the quality of heuristic algorithms proposed for the provably intractable cases. As well as this, we would like to investigate how different heuristic methods perform against each other upon the various realistic models we can implement for generating artificial social networks. These models exhibit particular properties that manifest themselves in the social networks that we observe in the real world. Typical examples would include Facebook, Tripadvisor, or Twitter.

In this section we explain the implementation of the rating-system that we use to perform our experiments upon. This involves the implementation of various random graph models for generating social networks, and the algorithms themselves. We will discuss the subtleties that we had to take into account when implementing the framework and our algorithms. The framework, algorithms, and random graph models have been implemented from scratch in Java.

A.1 Random Graph Models for Social Networks

In order to properly understand how information and influence diffuses through social networks, we have to have a model of the link structure within a society. Models for networks that exhibit desirable and realistic properties have long since been studied [18] [17] and we shall discuss two models that exhibit a range of varying properties. The motivation for doing so is so that we are able to construct appropriate customer networks upon which we can apply algorithms for computing bribing strategies and provide a quantitative analysis of their results. We shall also talk about realistic methods of representing individual customer evaluations within the network. As we shall see, the relative performance of our algorithms depends heavily on the properties of the network. These properties are essential to the feasibility of a fixed-parameter solution and a fine line between the tractability and intractability of such problems is clearly evident.

We implement models for generating two popular types of network that are said to be prevalent in real social networks: *scale-free* and *small-world* networks.

Definition A.1. *Scale-free network: A scale-free network is a network whose degree distribution exhibits (asymptotically) a power law. That is, let the fraction of nodes in the network having k connections to other nodes be $P(k)$. Then, $P(k) \sim k^{-\gamma}$.*

Definition A.2. *Small-world network: A small-world network is one where the distance d between two randomly chosen vertices grows proportionally to the logarithm of the number of vertices n or the network. That is, $d \propto log(n)$ [15].*

With regards to voter networks and in particular the rating system of [2], both of the above networks are useful to look at. We see extensive evidence in the literature that key properties of scale-free and small-world networks manifest themselves in voter networks. This is easy to see as one might expect, in some context, certain individuals to be *hubs* of a voter network. That is, there are a relatively large number of other individuals who are influenced by their opinion. This is just one characteristic of a scale-free network.

Alternatively, we see with small-world networks the regular occurrence of cliques or near cliques of individuals. This would be a very reasonable network characteristic to expect in reality in that it is often observed that the views and opinions amongst different demographics as groups of people are closely tied. A number of studies can be found, such as [18], as to the many different properties of voter networks.

A.1.1 Watts & Strogatz Model

A workhorse model for generating graphs with small-world properties is the *Watts & Strogatz* random graph model. The basic method of construction is to randomly rewire the edges of a regular lattice. An algorithm to construct such networks is as follows and is described in [15]. Let N denote the number of nodes, K the mean degree, and $\beta \in [0, 1]$.

1. Construct a regular ring lattice on N nodes by connecting each node to the $K/2$ neighbours either side of it.

2. For every node $n_i = n_0, .., n_N$ take every edge (n_i, n_j) with $i < j$ and rewire it with probability β. Rewiring is done by replacing some edge (n_i, n_j) with (n_i, n_k) where k is chosen with uniform probability over all possible values that avoid self-loops $(k \neq i)$ and link duplication. That is, if there already exists an edge we do not rewire as to duplicate that edge.

This model is used as one such method of randomly generating networks within the implementation.

A.1.2 Barabási-Albert Model

The Barabási-Albert model for generating random scale-free networks uses a *preferential attachment* mechanism. This means that the more connected a node is within a network, the higher the probability that it receives new links during construction. One such property of the resulting network is the presence of hubs. The basic algorithm for constructing Barabási-Albert networks is as described in [16].

1. We begin with an initial connected network on N nodes.

2. New nodes are added to the network one-by-one. A new node is connected to node i with probability

$$p_i = \frac{k_i}{\sum_j k_j}$$

   where $k_i = deg(i)$ and j runs over all pre-existing nodes.

A.1.3 Random Evaluation Assignment

Having constructed a voter network generated by one of the previously discussed random graph models, we can label vertices of the network with evaluations. The approximate desired proportion of all customers that should be made non-voters can be passed to the functions which generate networks. For example, we might pass a (realistic) value of 0.15

to enforce that in reality a large proportion of customers do express an opinion. Within the implementation, the way that this is (approximately) achieved is to firstly assign *every* customer a uniformly random evaluation between 0 and 1. If a customer's evaluation is below the threshold of the proportion of customers we desired to be non-voters (and every customer in its neighbourhood is influenced by at least another voter) we simply set the customer to being a non-voter. The result is that we end up with approximately the desired number of non-voters. This method of evaluation assignment works well in practice.

Remark A.1. *We might be tempted to implement a more realistic assignment of evaluations to voting customers than simply assigning them a uniformly random value. However, it is not the actual evaluations of customers that affect the computation of a bribing strategy, but the structure of the network (the number of non-voters) which is most influential. As explained, we can closely control this property.*

Bibliography

[1] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors. “Handbook of Computational Social Choice”. *Cambridge University Press* (2015).

[2] U. Grandi and P. Turrini. “A network-based rating system and its resistance to bribery”. *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (2016). pp. 301–307.

[3] Y. Chevaleyre et al. “A short introduction to computational social choice”. *International Conference on Current Trends in Theory and Practice of Computer Science.* Springer Berlin Heidelberg (2007).

[4] J. Bartholdi, C. Tovey, and M. Trick. “The computational difficulty of manipulating an election.” *Social Choice and Welfare 6.3* (1989). 227-241.

[5] C. Papadimitriou. “Computational complexity”. *John Wiley and Sons Ltd* (2003).

[6] D. Bovet and P. Crescenzi. “Introduction to the theory of complexity”. *Creative Commons* (2006).

[7] C. Lautemann. “BPP and the polynomial hierarchy”. *Information Processing Letters 17.4* (1983). 215–217.

[8] T. Cormen. “Introduction to algorithms”. *MIT press* (2009).

[9] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. “How hard is bribery in elections?”. *Journal of Artificial Intelligence Research 35* (2009). 485–532.

[10] D. Baumeister, E. Gábor, and R. Jörg. “How hard is it to bribe the judges? A study of the complexity of bribery in judgement aggregation”. *International Conference on Algorithmic Decision Theory. Springer Berlin Heidelberg* (2011).

[11] R. Bredereck et al. "A multivariate complexity analysis of lobbying in multiple referenda". *Journal of Artificial Intelligence Research 50* (2014). 409–446.

[12] R. Marti and G. Reinelt. "The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization". *Springer-Verlag Berlin Heidelberg* (2011). 978-3-642-16728-7.

[13] S. Young and E. Scheinerman. "Random dot product graph models for social networks". *Proceedings of the 5th International Conference on Algorithms and Models for the Web-graph* (2007). pp. 138–149.

[14] J. P. Onnela et al. "Structure and tie strengths in mobile communication networks". *Proceedings of the National Academy of Sciences 104.18* (2007). 7332-7336.

[15] D. J. Watts and S. H. Strogatz. "Collective dynamics of small-world networks". *Nature, 393(6684) 440442* (1998).

[16] A. Réka, and A. Barabási. "Statistical mechanics of complex networks". *Reviews of modern physics 74.1* (2002). 47.

[17] M. Newman, D. Watts, and S. Strogatz. "Random graph models of social networks". *Proceedings of the National Academy of Sciences 99. suppl 1* (2002). 2566-2572.

[18] M. Jackson. "Social and economic networks". *Princeton university press* (2010).

[19] M. Garey and D. Johnson. "Computers and intractability". *Vol. 29. New York: wh freeman* (2002).

[20] L. Fortnow. "The status of the P versus NP problem". *Communications of the ACM 52.9* (2009). 78-86.

[21] E. Losievskaja. "Approximation algorithms for independent set problems on hypergraphs" (2009).

[22] J. Flum and M. Grohe. "Parameterized Complexity Theory". *Volume XIV of Texts in Theoretical Computer Science. An EATCS Series* (2006).

[23] R. Diestel. "Graph theory". *Springer* (2000).

[24] C. Chekuri. "CS 598: Topics in Combinatorial Optimization", https://courses.engr.illinois.edu/cs598csc/sp2010/ [Last referenced 20/07/2017]. *University of Illinois* (2010).

[25] M. Cygan et al. “Dominating set is fixed parameter tractable in claw-free graphs”. *Theoretical Computer Science 412.50* (2011). 6982-7000.