Algorithmic Techniques for Influence Dynamics in Social Networks

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science

Giorgos Stamatelatos

Algorithms and Privacy Research Unit
Department of Electrical and Computer Engineering
Democritus University of Thrace

Advisor: Pavlos S. Efraimidis

Xanthi, July 2014

*Dedicated with extreme affection
and gratitude to my parents*

Acknowledgements

The work included in this thesis could not have been performed if not for the assistance, patience, and support of many individuals.

First and foremost I offer my sincerest gratitude to my advisor, Pavlos Efraimidis, who has supported me throughout my thesis with his patience, knowledge and expertise while allowing me the room to work in my own way. I attribute the level of my Master's degree to his encouragement and effort and without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier advisor.

Besides my advisor, I would like to thank the rest of my examining committee: Prof. Vassilis Tsaoussidis, Prof. Alexandros Karakos, Prof. Paul Spirakis and Assistant Prof. Spyros Kontogiannis for their insightful comments on this work.

I would also like to extend my appreciation to my colleague George Drosatos for his help and support during my studies. Finally, special thanks go to my friend Panos, aka "Takis", for his remarkable ability to solve seemingly complicated problems.

Giorgos Stamatelatos
Xanthi, July 2014

Abstract

Algorithmic techniques for influence dynamics in social networks

In this thesis, a non-cooperative, zero-sum game of influence between two firms on a social network is defined and examined. The social network is represented with a directed graph of n individual agents with an initial opinion on a particular subject. The DeGroot model defines the opinion dynamics and the rules of social interaction, according to which the belief of each individual is influenced by the opinions of their neighbors. The players of the game are two external entities who have diametrically opposed opinions. Each player tries to manipulate the community in their own opinion's favor, by actively interfering with the social graph structure. We investigate possible equilibria, identify basic principles for a successful strategy and compare the performance of several algorithms implementing player strategies.

Keywords – Social Networks, Opinion Dynamics, Influence Game

Περίληψη

Αλγοριθμικές τεχνικές για τη δυναμική επιρροής σε κοινωνικά δίκτυα

Η δυναμική επιρροής γνωμών είναι ένα σύνολο από μοντέλα που απορρέουν από τη μελέτη της προέλευσης και εξέλιξης των γλωσσών. Η δυναμική επιρροής στα κοινωνικά δίκτυα και η παιγνιοθεωρητική αντιμετώπιση τέτοιων θεμάτων πρόσφατα προσέλκυσε το ενδιαφέρον της ερευνητικής κοινότητας. Τέτοια μοντέλα αποτελούνται κατά κανόνα από έναν υψηλό αριθμό ατόμων, σε κάθε ένα από τους οποίους αποδίδεται μία κατάσταση που ονομάζεται *γνώμη*, την οποία ενημερώνουν μέσω ελληλεπιδράσεων με τους γείτονές τους. Σε τέτοια μοντέλα επιρροής, η συμπεριφορά των ατόμων ακολουθά έναν απλό κανόνα και η επίδραση της τοπολογίας του δικτύου λαμβάνεται υπ' όψην μέσω των αλληλεπιδράσεων λόγω γειτνίασης. Ως εκ τούτου, τα μοντέλα κοινωνικής επιρροής παρέχουν εργαλεία για την προσομοίωση της συλλογικής συμπεριφοράς σε πολύπλοκα συστήματα της κοινωνιολογίας, φυσικής και επιστήμης των υπολογιστών.

Στην μεταπτυχιακή αυτή εργασία ορίζεται ένα ανταγωνιστικό παίγνιο επιρροής πάνω σε κοινωνικά δίκτυα ενώ ταυτόχρονα εξετάζονται πιθανά σημεία ισορροπίας του καθώς και αποδοτικές στρατηγικές για τους δύο παίκτες.

Το κοινωνικό δίκτυο πάνω στο οποίο έχει οριστεί το παίγνιο μοντελοποιείται με μορφή γραφήματος, του οποίου οι κορυφές αντιπροσωπεύουν πρόσωπα και οι ακμές σχέσεις επιρροής μεταξύ των προσώπων. Το κοινωνικό δίκτυο υπακούει στους κανόνες του μοντέλου DeGroot, σύμφωνα με το οποίο οι κόμβοι του κοινωνικού δικτύου προσαρμόζουν τη γνώμη τους (για λόγους απλότητας ένας δεκαδικός αριθμός μεταξύ 0 και 1) ανάλογα με τις γνώμες των κόμβων με τους οποίους γειτνιάζουν με επαναληπτικό τρόπο, φαινόμενο που οδηγεί (υπό ορισμένες προϋποθέσεις) σε ισορροπία των γνωμών. Για τους σκοπούς της εργασίας χρησιμοποιούνται απλά γραφήματα, όπως path, cycle, wheel, αλλά και γραφήματα scale-free. Τα τελευταία πλησιάζουν περισσότερο μορφολογικά σε ένα πραγματικό κοινωνικό δίκτυο.

Στο παραπάνω δίκτυο προστίθενται δύο επιπλέον κορυφές (P_0 και P_1 που αποτελούν τους δύο παίκτες του παιγνίου) με γνώμες 0 και 1 αντίστοιχα. Οι επιπλέον κορυφές έχουν το χαρακτηριστικό ότι δεν επηρεάζονται από άλλον κόμβο του δικτύου (άρα σύμφωνα με το μοντέλο DeGroot η γνώμη τους παραμένει σταθερή) αλλά επηρεάζουν άλλους κόμβους. Μετά την εκτέλεση του επαναληπτικού υπολογισμού DeGroot, ο μέσος όρος όλων των γνωμών του γραφήματος θα κρίνει ποιος εκ των P_0 και P_1 υπερίσχυσε, ανάλογα με το αν αυτός είναι μεγαλύτερος ή μικρότερος από το $1/2$. Στην εργασία αναλύονται σενάρια στα οποία κάθε παίκτης έχει περιορισμένους πόρους, υπάρχει δηλαδή μέγιστο όριο στις κορυφές που μπορεί να επηρεάσει.

Από τα παραπάνω γίνεται σαφές ότι στόχος κάθε παίκτη είναι να επιλέξει τις καταλληλότερες κορυφές (στο εξής “κίνηση”) ώστε να κερδίσει τον αντίπαλο παίκτη. Για την εύρεση της βέλτιστης κίνησης σε απλά γραφήματα και για την κατανόηση του παιγνίου χρησιμοποιείται μια επαναληπτική brute-force τεχνική που σε κάθε της βήμα προσπαθεί να βρει μια καλύτερη κίνηση από την προηγούμενη καλύτερη. Με τη μέθοδο αυτή γίνεται σαφές ότι η στρατηγική ενός παίκτη πρέπει να ικανοποιεί δύο συνθήκες ώστε να υπερισχύσει του αντιπάλου του: η κίνησή του να αποτελείται από κορυφές με υψηλό eigenvector centrality ενώ ταυτόχρονα θα πρέπει αυτές να είναι διεσπαρμένες σε όλο το δίκτυο και όχι συγκεντρωμένες σε κοντινά σημεία. Η παρατήρηση αυτή οδηγεί σε χρήση heuristic αλγορίθμων, όπως του k-center για μια καλή λύση σε πολύπλοκα γραφήματα, όπου είναι αδύνατη η χρήση brute force τεχνικών λόγω πολυπλοκότητας.

Στα πλαίσια της εργασίας έχει αναπτυχθεί μια συλλογή εργαλείων σε java, πάνω στην οποία έχει γίνει πλήθος δοκιμών και πειραμάτων.

Λέξεις-Κλειδιά – Κοινωνικά Δίκτυα, Δυναμική Επιρροής, Παίγνιο Επιρροής

Contents

Abstract **vii**

Περίληψη **ix**

Contents **xi**

Symbols **xiii**

1 Introduction **1**

2 Background **3**

2.1 The Social Graph 3

2.1.1 The Adjacency Matrix 3

2.1.2 Definitions 4

2.1.3 Centrality 5

2.2 The DeGroot Model 5

2.2.1 Updating Process 6

2.2.2 Convergence 7

2.2.3 Google's PageRank 8

3 The Game-Theoretic Model **11**

3.1 Definition 11

3.1.1 Empirical Definition 11

3.1.2 Formal Definition 11

3.2 Examples 12

3.2.1 First Example 12

3.2.2 Second Example 14

3.3 The Challenge 15
3.4 Strategy Principles 16

4 Algorithms **17**
4.1 Random Brute-Force Search Algorithm 17
4.2 Greedy Algorithm 19
4.2.1 Algorithm Process 19
4.2.2 Example 20
4.2.3 Strategy Principles 22
4.3 Generalized Greedy Algorithm 23

5 Experimental Results **25**
5.1 Path Graph 26
5.2 Cycle Graph 28
5.3 Two-Wheels Graph 30
5.4 Scale-Free Graph 31
5.5 Scale-Free Cluster Graph 33

6 Discussion **37**
6.1 Strategy Principles 37
6.2 Algorithm Conclusions 37
6.3 Future Work 38

A Figure and Diagram Sources **41**

Bibliography **45**

Symbols

d	damping factor
d_{avg}	average degree
d_i	degree of node N_i
D	density
D_{ij}	distance map
E	number of edges
m	number of move nodes
n	number of nodes
N_i	the node with ID i
p_i	opinion of N_i
P_0, P_1	the two players
r_i	PageRank of N_i
s	eigenvector centrality
$score$	the final state of the game in $[0, 1]$
S	suggested move
T_{ij}	adjacency matrix
W_i	weight of index i

Chapter 1

Introduction

The **opinion dynamics** is a class of agent based models arising from the study of the origins and evolutions of languages. Influence dynamics in social networks and game-theoretic treatment of such problems has recently attracted the interest of the research community (see for example the books [11, 12] or the papers [6, 8, 10, 19]). These models are typically consisted of a large number of individuals, each of which is assigned with a state called *opinion* and updates its opinion through interactions with its neighbors. In these opinion dynamics models, the individual behavior follows a very simple rule and the effect of the network topology is introduced by the neighbor interactions, so these models provide powerful tools to simulate and investigate the collective behavior in complex systems in sociology, physics and computer science.

In this thesis we focus on the **DeGroot model**, according to which the belief of each individual is constantly influenced by the opinions of their neighbors until the network reaches an agreement (consensus). An interesting aspect that derives from this model is the concept of **stubborn agents** and the influence they can exert over a social community. Stubborn agents with opposing views prevent consensus. This behavior raises further questions: How can a stubborn agent affect the overall opinion of the society in his favor? How can a stubborn agent prevail over another one with opposing belief? What is the best way to impose one's view over a human society? These types of phenomena are seen in major social fields, like politics and economics.

In this work, we will try to answer these questions by defining an **influence game** between two firms on a social network. The social network is represented with a graph; each vertex represents an individual and each edge represents a form of trust or influence. We will examine why stubborn agents prevent consensus using the rules dictated by the DeGroot model. Furthermore, we will establish some basic principles that define a successful strategy for a stubborn agent to optimally impose his opinion over the society. Finally, we will refer to two algorithms that we developed, one based on a brute-force and one on a greedy method, and perform simulations for those agents on various types of graphs.

The rest of this thesis is structured as follows:

Chapter 2: Background

In this chapter, we provide the necessary background for understanding key concepts in this thesis. Readers familiar with graph theory and the DeGroot model may skip this chapter.

Chapter 3: The Game-Theoretic Model

The game is being defined and its mechanics are examined. We provide examples for better understanding as well as graph drawings. In this chapter we will also refer to a few important principles that make up a decent strategic behavior.

Chapter 4: Algorithms

Taking into consideration the principles of Chapter 3, we analyze two algorithms that we developed for computing action suggestions: one based on a brute-force method (random brute-force search) and one based on a heuristic (greedy algorithm).

Chapter 5: Results

In this chapter, the results of our tests are presented in the form of tables. The experiments are separated based on the graph type. Since in many problem instances we are not aware of the optimal strategy, we evaluate our algorithms by comparing them with each other and specifically within a round-robin tournament context.

Chapter 6: Discussion

Finally, we will summarize our work, recap our conclusions and refer to open problems and aspects that require further examination.

Chapter 2

Background

In this chapter, we will refer to some preliminary concepts on social graphs and graph theory as well as the DeGroot model, which defines the opinion dynamics and the rules of social interaction in a community.

Two of the most successful and recent studies for social networks and game theory are Jackson [11] and Easley and Kleinberg [12].

2.1 The Social Graph

Let $N = \{1, 2, \ldots, n\}$ be the set of nodes that are involved in a network of relationships. Nodes will also be referred to as vertices, individuals or agents, depending on the setting. It is important to emphasize that nodes might be individual people, countries, or other organizations; or a node might even be something like a web page belonging to some person or organization.

The network form -as being discussed in this thesis- is a weighted directed graph, where one node i is connected to another node j or not. Furthermore a label (weight) is associated with every edge in the graph. Finally, there can't be more than one edge from a node i to another node j. It is worth mentioning that there are social communities or even Internet social networks that a joint consent is needed to establish a relationship, for example Facebook which can be represented with an undirected graph (whereas the "follower" mechanic of Twitter creates a directed graph). However, from a mathematical perspective, the concept of a directed graph is broader, hence this is what we are going to be based on.

2.1.1 The Adjacency Matrix

An adjacency matrix is a means of representing which vertices of a graph are adjacent to which other vertices. Specifically, the adjacency matrix of a graph G on n vertices is the $n \times n$ matrix where the non-diagonal entry a_{ij} is the weight of the edge from vertex i to vertex j, and the diagonal entry a_{ii} is the weight of the edge (loop) from vertex i to itself. The normalized adjacency matrix, which will be referred later, is the adjacency matrix whose rows sum to 1.

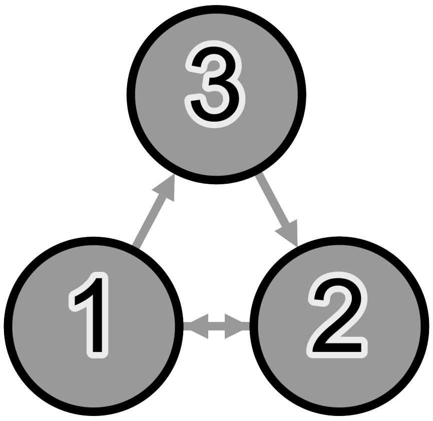

FIGURE 2.1: An example graph with 3 nodes.

As an example, the adjacency matrix of the graph in Figure 2.1 (assuming all edge weights are equal) is

$$T = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.1}$$

while the normalized adjacency matrix is

$$T_n = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

2.1.2 Definitions

Strongly connected – A directed graph is strongly connected if every vertex is reachable from every other following the directions of the arcs (edges). For example, the graph in Figure 2.1 is strongly connected.

Diameter – The distance between two nodes is the length of the shortest path or geodesic between them. If there is no path between the nodes, then the distance between them is infinite. The diameter of a network is the largest distance between any two nodes in the network [1]. Any strongly connected directed graph has finite diameter.

Degree – The degree d_i of a node i is the number of links that involve that node. For a directed graph, two measures exist: *in-degree* and *out-degree*. In the context of this thesis, we will refer to *in-degree* simply as *degree*. For undirected graphs, the average degree is $d_{avg} = {}^{2E}/_{n}$.

Density – The density D of a network keeps track of the relative fraction of links that are present, and for an undirected graph is defined as ${}^{E}/_{E_{max}}$, where E is the number of edges on the network and E_{max} the maximum number of edges. E_{max} is, by definition, equal to ${}^{n(n-1)}/_{2}$, so $D = {}^{2E}/_{n(n-1)} = {}^{d_{avg}}/_{(n-1)}$.

2.1.3 Centrality

Centrality refers to indicators which identify the most important vertices within a graph. Applications include identifying the most influential person(s) in a social network, key infrastructure nodes in the Internet or urban networks, and super spreaders of disease. Centrality concepts were first developed in social network analysis, and many of the terms used to measure centrality reflect their sociological origin. [14]

Degree centrality – Perhaps the simplest measure of the importance of a given node in a network is its degree. A node with degree $n-1$ would be directly connected to all other nodes, and hence quite central to the network. A node connected to only 2 other nodes (assuming large n) would be, at least in one sense, less central. The degree centrality of a node i is normalized in $[0, 1]$ by dividing its degree d_i with $n - 1$ and tells us how well it is connected, in terms of direct connections. [11]

Closeness centrality – In connected graphs there is a natural distance metric between all pairs of nodes, defined by the length of their shortest paths. The farness of a node s is defined as the sum of its distances to all other nodes, and its closeness is defined as the inverse of the farness. [16]

Betweenness centrality – Betweenness centrality of a node s is equal to the number of shortest paths from all vertices to all others that pass through s. It was first proposed by Freeman, who has also developed a number of other centrality measures. [7]

In the next section (Section 2.2), we will refer to additional centrality measures that are related to the unit-eigenvector of the adjacency matrix T.

2.2 The DeGroot Model

The seminal network interaction model of information transmission, opinion formation, and consensus formation is due to DeGroot. [5]

2.2.1 Updating Process

Individuals in a society start with initial opinions on a subject. Let these be represented by a n-dimensional vector of probabilities

$$p(0) = \begin{bmatrix} p_1(0) & p_2(0) & \dots & p_n(0) \end{bmatrix}^{\mathrm{T}}.$$

Each $p_i(0)$ lies in the interval $[0, 1]$ and might be thought of as the probability that a given statement is true, or the quality of a given product, or the likelihood that the individual might engage in a given activity etc. The interaction patterns are captured through the normalized adjacency matrix T mentioned in Section 2.1.1. The interpretation of T_{ij} is that it represents the weight or trust that agent i places on the current belief of agent j in forming his or her belief for the next period. Beliefs are updated over time so that

$$p(t) = T\,p(t-1) = T^t\,p(0) \tag{2.3}$$

For the example of Figure 2.1, the normalized adjacency matrix is

$$T = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.4}$$

Suppose that we begin with a vector of beliefs given by

$$p(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.5}$$

According to the updating rule in Formula 2.3

$$p(1) = Tp(0) = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.6}$$

Then, as agents update again, beliefs become

$$p(2) = Tp(1) = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0 \\ 1 \end{bmatrix} \tag{2.7}$$

Iterating this process leads to beliefs that converge:

$$\lim_{t\to\infty} p(t) = \begin{bmatrix} 2/5 \\ 2/5 \\ 2/5 \end{bmatrix} \tag{2.8}$$

2.2.2 Convergence

As illustrated in the previous example, the updating process converges if

$$\lim_{t\to\infty} p(t) = \lim_{t\to\infty} T^t p(0) \tag{2.9}$$

exists for all initial vectors of beliefs $p(0)$. In order for this to happen, T^t must converge. In the above example:

$$\lim_{t\to\infty} T^t = \begin{bmatrix} 2/5 & 2/5 & 1/5 \\ 2/5 & 2/5 & 1/5 \\ 2/5 & 2/5 & 1/5 \end{bmatrix} \tag{2.10}$$

Thus,

$$\lim_{t\to\infty} p(t) = \lim_{t\to\infty} T^t p(0) = \begin{bmatrix} 2/5 & 2/5 & 1/5 \\ 2/5 & 2/5 & 1/5 \\ 2/5 & 2/5 & 1/5 \end{bmatrix} \times \begin{bmatrix} p_1(0) \\ p_2(0) \\ p_3(0) \end{bmatrix} \tag{2.11}$$

$$= \left(\frac{2}{5}p_1(0) + \frac{2}{5}p_2(0) + \frac{1}{5}p_3(0)\right)\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}} \tag{2.12}$$

Hence, no matter what beliefs $p(0)$ the agents start with, they all end up with the same limiting beliefs. The example also illustrates that agents 1 and 2 have twice as much influence over the limiting beliefs as agent 3 does.

Not all graphs converge, however. For instance, a graph with adjacency matrix

$$T = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.13}$$

will oscillate

$$T^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 \end{bmatrix}, T^3 = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, T^4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 \end{bmatrix} \tag{2.14}$$

and there is no convergence. According to Golub and Jackson [9, Theorem 2], a stochastic matrix T is convergent if and only if it is strongly aperiodic.

This process can be generalized to any graph that has convergence (assuming $n = 3$ for simplicity):

$$\lim_{t\to\infty} T^t = \begin{bmatrix} s_1 & s_2 & s_3 \\ s_1 & s_2 & s_3 \\ s_1 & s_2 & s_3 \end{bmatrix} = s \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}, \text{ where } s = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix} \tag{2.15}$$

We already know that

$$\lim_{t\to\infty} p_i(t) = s \times p(0), \text{ for any } p(0) \tag{2.16}$$

Since this is true for any $p(0)$, it must also be true for $p(1)$. Thus,

$$s \times p(0) = s \times p(1) \Rightarrow s \times p(0) = s \times (T \times p(0)) \tag{2.17}$$

And since this, again, has to hold for any $p(0)$, it follows that

$$sT = s \tag{2.18}$$

Thus, s is a left-hand eigenvector of T corresponding to the eigenvalue 1 and is defined as the **eigenvector centrality** of the graph. [11, Section 8.3.5]

2.2.3 Google's PageRank

Social influence, as defined in the context of the DeGroot model provides a foundation for eigenvector based centrality measures. It also provides a basis for understanding other related systems. In particular, the structure of Google's PageRank system [15] is analogous to the influence vectors here, where the T matrix is derived by normalizing the directed links between web pages (so that $T_{ij} = {}^{1}/_{d_i}$ if page i has a link to page j, and d_i is the number of directed (out) links that page i has to other pages). In a graph with normalized adjacency matrix T, the calculation of PageRank is an iterative process based on the following rule:

$$r(t) = d + (1-d)\, T^{\mathrm{T}} r(t-1) \tag{2.19}$$

where T^{T} is the transpose adjacency matrix, $r(0)$ is a vector of ones and d a parameter called damping factor. If $d = 0$ then the updating rule becomes

$$r(t) = T^{\mathrm{T}} r(t-1) \tag{2.20}$$

Thus, when convergence is achieved

$$\lim_{t\to\infty} r(t) = \lim_{t\to\infty} \left(T^{\mathrm{T}}\right)^{t} r(0) = \left(\lim_{t\to\infty} T^{t}\right)^{\mathrm{T}} r(0) \tag{2.21}$$

because $(AB)^{\mathrm{T}} = B^{\mathrm{T}}A^{\mathrm{T}} \Rightarrow \left(A^2\right)^{\mathrm{T}} = \left(A^{\mathrm{T}}\right)^2$. But since $r(0)$ is a vector of ones, according to Equation 2.15 we have that

$$\lim_{t\to\infty} r(t) = s \tag{2.22}$$

The above equation depicts the fact that PageRank with $d = 0$ is in fact the eigenvector centrality.

Chapter 3

The Game-Theoretic Model

3.1 Definition

3.1.1 Empirical Definition

The game proposed in this thesis comprises a social network of n individual agents with an initial opinion on a particular subject. Two external individuals (the players) approach with the task of manipulating the community in their own opinion's favor. These two agents have diametrically opposed opinions and will not listen to anyone in such a way that their opinion on that matter remains unmodified. Each player chooses a set of preexisting agents and tries to "talk them out", affecting their opinion, which in turn affects the opinion of their neighbors and so on. The players have limited resources in order to carry out their task: they may "approach" at most a certain number of other agents and they also have limited "time" in terms of completing their whole objective.

3.1.2 Formal Definition

This non-cooperative[1] game consists of a graph, two players and their respective moves and terminates upon DeGroot convergence.

The graph – This is the graph that represents the social network. The graph acts similarly to a "map" in a strategy video game[2]; players may adapt their strategy depending on the map. There are n vertices in the graph, all of which have initial opinions of 0.5 (though the initial opinion vector doesn't matter as we will later demonstrate).

[1]http://en.wikipedia.org/wiki/Non-cooperative_game
[2]http://en.wikipedia.org/wiki/Real-time_strategy

The players – The two adversaries are represented by two additional vertices on the graph (from now on "player vertices" or "stubborn agents") with initial opinions 0 and 1 (let them be P_0 and P_1). Since these vertices' opinions remain unchanged, they don't have any outbound edges. This type of node behavior can be expressed in our model with a self-link (or else a loop) with (normalized) weight 1.

The player moves – A player move or a player action is a group of vertices that the respective player wishes to directly influence (the "move nodes" or "action vertices"). The move (eg. for P_0) is a vector $[(N_1 = W_1), (N_2 = W_2), \dots, (N_m = W_m)]$ modeled in the form of edges with target P_0 and sources N_1, N_2, ..., N_m with weights W_1, W_2, ..., W_m. Certain restrictions were applied on a move, like the maximum amount of nodes to influence (m) and the maximum sum of weights ($W_1 + W_2 + \dots + W_m$).

Convergence – We consider convergence on the DeGroot model to be the final state of the game. The average opinion value (*score*) determines the winner according to a simple formula: The player P_0 wins if the average opinion of the network is less than 0.5, otherwise P_1 wins. A draw as an outcome is very uncommon and usually only occurs when the actions from both players are identical or if the graph's geometry is symmetric.

The utility function of P_1 is $U_1 = score$ because P_1 is trying to shift the average opinion of the network to 1 and the utility function of P_0 is $U_0 = 1 - score$. Since the sum of the utility functions is constant, this is a zero-sum game[3]. It is worth noting that swapping the opinions of the two players will result in a complementary average opinion $(1 - score)$ (this will become more clear towards the end of Section 3.2.1), thus the value of the opinion itself (0 or 1) on the stubborn agents does not matter.

3.2 Examples

3.2.1 First Example

We will now use the graph in Figure 2.1 to demonstrate the game. We will also define the following parameters

$$m = 1 \tag{3.1}$$

$$W_1 = 1 \tag{3.2}$$

[3] http://en.wikipedia.org/wiki/Zero-sum_game

meaning that each of the two players can affect only one node with weight 1[4]. During the game, P_0 selects N_3 and P_1 selects N_1. According to the definition we gave earlier, the graph becomes the one illustrated in Figure 3.1.

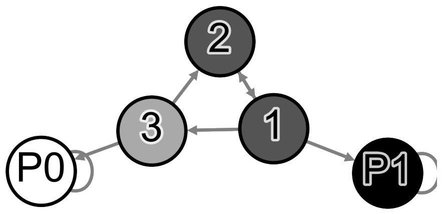

FIGURE 3.1: An example graph of a game. Nodes are colored by the final beliefs.

Vertices in this figure are already colored by their final opinion: black represents opinion 1 and white represents opinion 0. Before we even dive in more detailed description, it is being clear from the figure itself that nodes "close" to P_1 have darker color. The opposite stands true for P_0. The adjacency matrix of this graph (including the stubborn agents and assuming P_0 is the fourth row and P_1 is the fifth row) is

$$T = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The normalized version of this matrix is

$$T_n = \begin{bmatrix} 0 & 1/3 & 1/3 & 0 & 1/3 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

It can be easily shown that

$$\lim_{t\to\infty} T_n^t = \begin{bmatrix} 0 & 0 & 0 & 1/3 & 2/3 \\ 0 & 0 & 0 & 1/3 & 2/3 \\ 0 & 0 & 0 & 2/3 & 1/3 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

[4]Obviously, these are the maximum values. However, there is no reason for a player not to fully exploit them.

So, assuming an initial opinion vector

$$p(0) = \begin{bmatrix} p_1(0) & p_2(0) & p_3(0) & 0 & 1 \end{bmatrix}^{\mathrm{T}} \tag{3.6}$$

the limiting beliefs of the network are given by

$$\lim_{t\to\infty} p(t) = \left(\lim_{t\to\infty} T_n^t\right) p(0) = \begin{bmatrix} 2/3 & 2/3 & 1/3 & 0 & 1 \end{bmatrix}^{\mathrm{T}} \tag{3.7}$$

The average opinion on this final state is $^{8}\!/_{15}$ so P_1 is the winner. We have shown that $p(0)$ does not play any role in the outcome of the game (as long as the initial beliefs of the stubborn agents are 0 and 1). The element ij of the matrix in Equation 3.5 represents how much influence does j exert over i.

One can notice that the final average opinion vector is the last column of the matrix of Equation 3.5. As such, if the opinions of the two players are swapped (or the players themselves), the final average opinion vector will be the second last column of that matrix, which is complementary to the last since the matrix is row stochastic.

3.2.2 Second Example

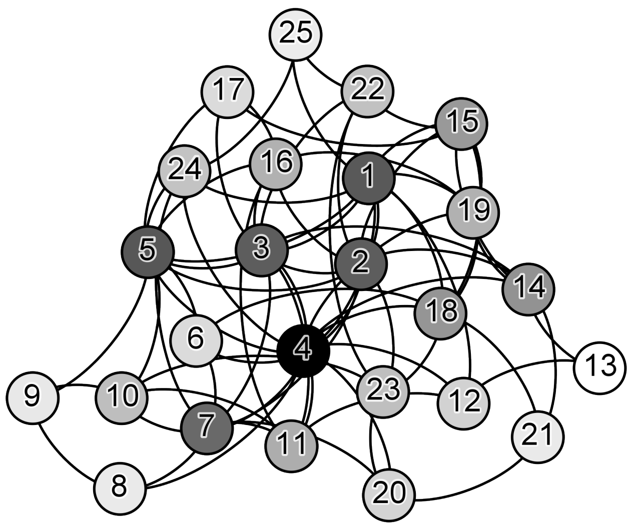

FIGURE 3.2: A scale-free graph with 25 nodes. Vertices are colored by their PageRank value.

Let's take a look at a more complicated example illustrated in Figure 3.2. This time, we define

$$m = 3 \tag{3.8}$$

$$\sum_{i=1}^{m} W_i = 3 \tag{3.9}$$

Each player can influence up to $m = 3$ nodes and the total weight of each player action cannot exceed 3. Suppose P_0 selects the move

$$[1 = 1, 6 = 1, 15 = 1]$$

and P_1 selects the move

$$[7 = 1, 17 = 1, 22 = 1]$$

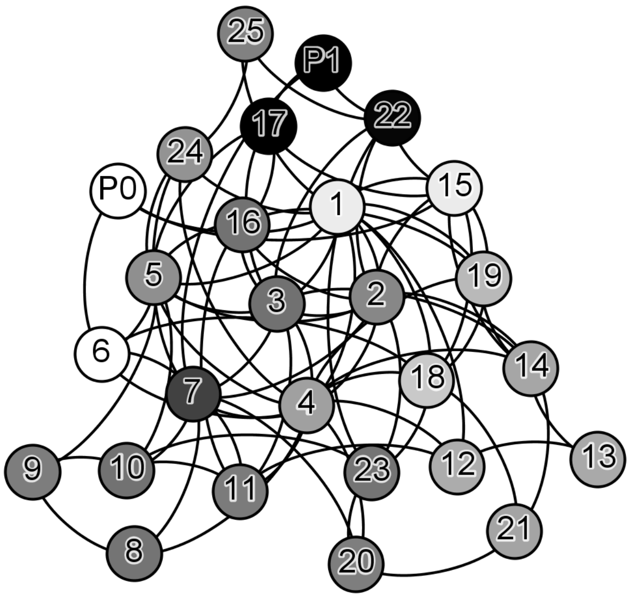

FIGURE 3.3: A game with the scale-free graph of Figure 3.2. Vertices are colored by their final belief value in a white-black scale.

The average opinion after convergence becomes approximately 0.4981 and P_0 is the winner. The graph of the network after convergence is illustrated in Figure 3.3 along with the limiting beliefs colored in a white-black scale.

3.3 The Challenge

Our challenge is to develop an algorithm for computing the optimal player action or at least a very good approximation of it; the more the action shifts the average limiting belief of the social network towards the player's opinion, the better the move is. In order to achieve this, we need to state some major principles that should characterize the strategic behavior of a player.

3.4 Strategy Principles

The following principles are based on observations that were made during experiments and tests. Some points are also justified mathematically up to an extent. It is important to mention that not all of these principles apply to all types of graphs but rather provide some general insight for a successful strategy.

1. The action must be spread over the network in order for the action vertices to be close to the majority of the graph nodes. It is unwise for a player to select too few nodes or nodes that are close to each other and don't span over the entire graph. This is very reasonable since the stubborn opinion tends to weaken as the distance from the player vertex increases creating a normal-like distribution around each of the directly influenced nodes. High amount of spanning will result in less redundancy because the opinion of an individual cannot exceed 1.

2. The action should contain nodes with a relatively high centrality measure (for example PageRank or eigenvector centrality). It is also intuitive that directly influencing a node that has already a high amount of influence over others will have greater overall impact on the network according to the DeGroot model since a vertex contribution to the overall opinion of the graph depends on the respective column of the adjacency matrix.

3. The move must exhaust its limits regarding the number of nodes to influence. A move with m vertices will almost always and on any graph or any circumstances beat a move with less than m vertices.

Chapter 4

Algorithms

Based on the strategy principles, we examined two algorithms whose aim is the suggestion of a decent player action and developed the first one based on a brute-force approach (random brute-force search) and the other one using a greedy, set-covering technique.

4.1 Random Brute-Force Search Algorithm

The brute-force method that was developed is a local search approach that aims in the exhaustive search of moves in order to find the optimal one, relying on the computational power to solve the problem. While a brute-force search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions – which in our case tends to grow very quickly as the size of the move vector (the move nodes) increases. Therefore, the brute-force method is used when the problem size is limited (for example, on a small graph) or for instructional purposes or as a means of understanding the underlying problem and noting basic principles for its solution.

Algorithm Principle – The brute-force is an iterative method that initially starts with an arbitrary move C and on each step picks a random move (random vertices with random weights) R. C and R are then simulated as two opponents and the winning move replaces C. Due to the fact that this algorithm can potentially run forever, a time limit parameter is included. Figure 4.1 illustrates this algorithm.

The total number of possible non-weighted solutions is

$$\gamma = C(n, m) = \frac{n!}{m!\,(n-m)!} \tag{4.1}$$

i.e. the number of combinations of m allowed number of vertices per move out of n vertices residing in the network. The expected number of random choices of the algorithm until all

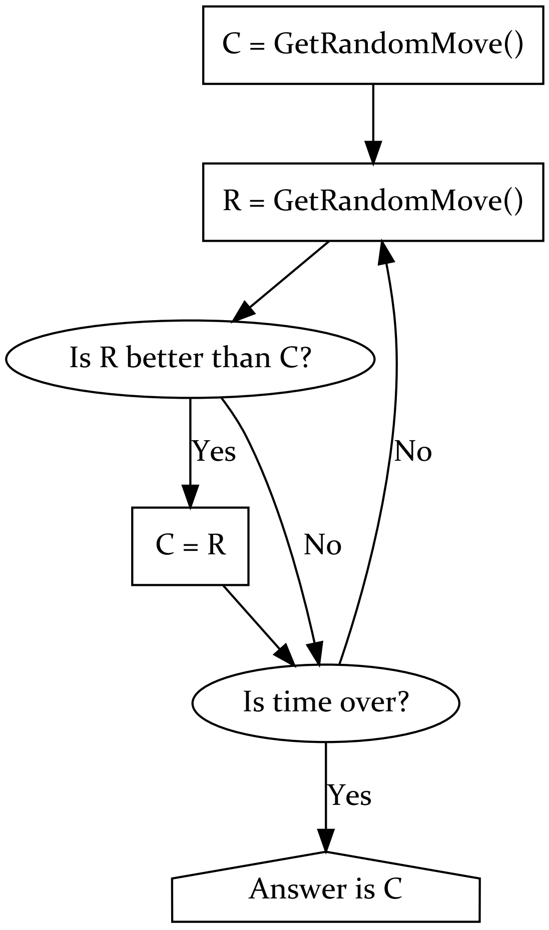

FIGURE 4.1: Diagram of the random brute-force search algorithm.

possible solutions are examined is $O(\gamma \cdot log(\gamma))$. Thus, when n is constant the complexity of this algorithm increases dramatically in relation to m (up to $n/2$) or if m is fixed, the complexity drastically increases as the number of vertices n increases.

Despite brute-force algorithm's complexity, in practice it will quickly find a relatively decent move without the need of actually simulating every possible combination of moves. In fact, as we will later observe in Chapter 5, doubling the amount of time that the brute-force algorithm is allowed to run, results in only minimal –but not negligible– gains in performance. This is due to the fact that it gets progressively harder to achieve a better move once a decent one has been established. In other words, the "goodput" of the algorithm drops over time.

In some rare cases, brute-force algorithm will be trapped on a loop; there exist a set of moves $[C_1, C_2, \ldots, C_p]$ at which the algorithm is trapped and oscillates. For example, given a set of three moves $[A, B, C]$, A beats B which is turn beats C, but C beats A. Based on the Diagram 4.1, the algorithm will be trapped on that set and in such case, a random move from the set will be suggested. The explanation seems to be that these problem instances do not have pure Nash equilibria. This scenario resembles the rock-paper-scissors game[1]. Often times, the algorithm will completely escape the loop and other times it will escape the set but get trapped in another one. Studying these aspects of the influence game is outside the scope of this thesis, but might be an interesting direction for future work.

4.2 Greedy Algorithm

As expected, the randomized brute-force search approach becomes cumbersome and unresponsive in large graphs. Thus, we developed a greedy algorithm that is only executed in a limited number of steps and -in our tests- yielded a very good approximation of the actual solution and in some cases a precisely optimal move, mainly on the smaller graphs.

4.2.1 Algorithm Process

The algorithm initially creates a vertex distance map, which is a matrix D. D_{ij} specifies the shortest path distance from node j to node i. Afterwards, the matrix F_{ij} is created such that

$$F_{ij} = \frac{1}{e^{D_{ij}}} \tag{4.2}$$

A vector Q of size equal to the number of vertices in the graph is also initialized with ones. The greedy algorithm will perform exactly m steps, where m is the allowed number of vertices per move, and on each step a node N_i will be selected based on a heuristic. The vector

$$S = \left[\left(N_1, {}^{1}\!/\!_{m}\right), \left(N_2, {}^{1}\!/\!_{m}\right), \ldots, \left(N_m, {}^{1}\!/\!_{m}\right)\right] \tag{4.3}$$

is the suggested move and the output of the algorithm. The algorithm does not take into consideration move weights so all move nodes have equal weight. At each step of the operation, the vector Θ is computed as:

$$\Theta = (1 - F) \times Q \tag{4.4}$$

and the x is selected as the index of the minimum element of Θ. N_x will then be added to the vector S. Finally, a new Q is computed as

$$Q_{new} = (1 - F_x)^{\mathrm{T}} \cdot Q \tag{4.5}$$

where F_x is the x-th row of F. The operation is repeated m times, at which point the vector S contains exactly m elements. This process is illustrated in Figure 4.2.

[1]http://en.wikipedia.org/wiki/Rock-paper-scissors

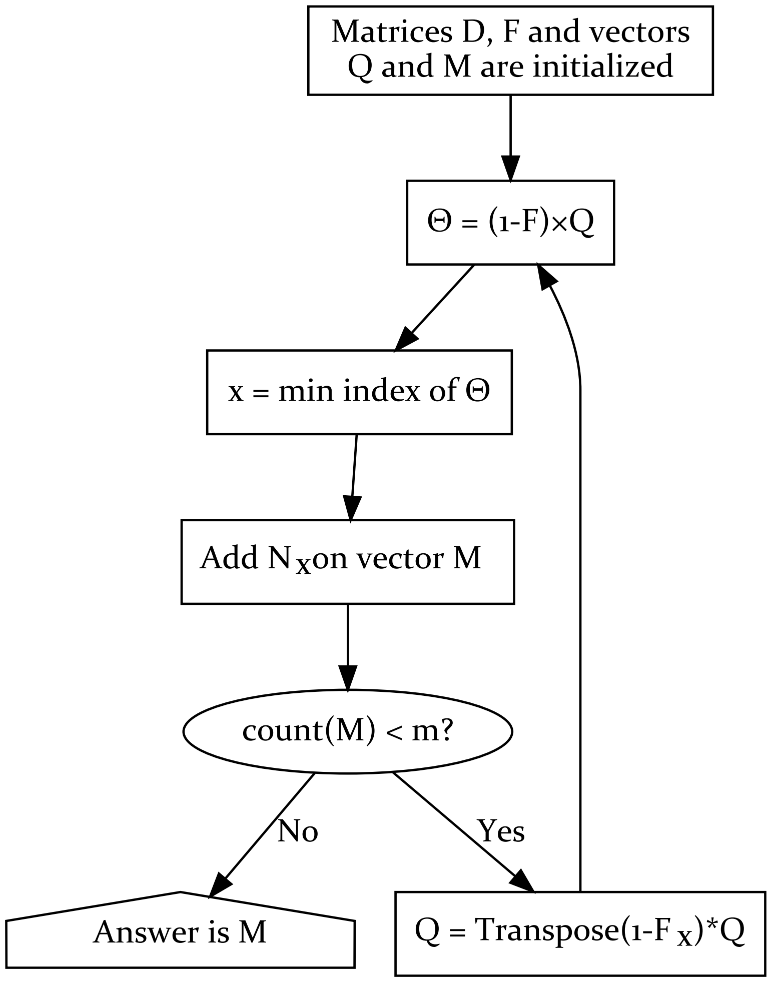

FIGURE 4.2: Diagram of the greedy algorithm.

4.2.2 Example

We will now demonstrate the execution of this algorithm in the example graph of Figure 2.1 (also included in this section for convenience) with 2 nodes per move. Some basic measures of this graph are:

- **Eigenvector centrality** – $\begin{bmatrix}1.2 & 1.2 & 0.6\end{bmatrix}$
- **PageRank** – approx. $\begin{bmatrix}1.163 & 1.192 & 0.644\end{bmatrix}$
- **Diameter** – 2

The (normalized) adjacency matrix T of this graph is

$$T = \begin{bmatrix}0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0\end{bmatrix} \tag{4.6}$$

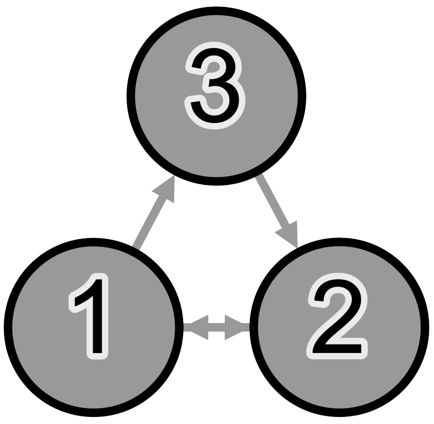

FIGURE 4.3: An example graph with 3 nodes.

Initially, the greedy algorithm, creates the distance matrix D and the matrix F:

$$D = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix} \Rightarrow F = \begin{bmatrix} 1 & 1/e & 1/e^2 \\ 1/e & 1 & 1/e \\ 1/e & 1/e^2 & 1 \end{bmatrix} \tag{4.7}$$

The element F_{ij} implies a certain amount of trust from agent j to agent i. It is not a measure of influence since it is only affected by the shortest path between them but rather an approximation. Furthermore, we initialize the vector Q:

$$Q = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{4.8}$$

Vector Q (quota) represents a measure that shows how much "entropy" a vertex has. The purpose of this algorithm is to reduce the overall "entropy"; to minimize the sum of vector Q. Notice that we do not take into consideration edge weights on this algorithm. All edges have the same amount of weight.

Every time a vertex (for example E) is selected by the algorithm, Q is adjusted in such a way that nodes close to E have their quota reduced more than those away from E. The exact formula of quota reduction is

$$Q_i = Q_i \cdot (1 - F_{Ei}) \tag{4.9}$$

This is equivalent to Formula 4.5 for all i's.

In order to find the best move with 2 nodes we will execute exactly 2 steps of the algorithm. In each step we will try to find which node minimizes the sum of Q. So, in the first step we will test all 3 nodes, as to which minimizes Q. In the case where we select the node with ID 1 for example, the sum of vector Q after the adjustment will be (based on Formula 4.9)

$$\Theta_1 = \sum_{i=1}^{3} Q_i \cdot (1 - F_{1i}) \tag{4.10}$$

Similarly, for nodes 2 and 3:

$$\Theta_2 = \sum_{i=1}^{3} Q_i \cdot (1 - F_{2i}) \tag{4.11}$$

$$\Theta_3 = \sum_{i=1}^{3} Q_i \cdot (1 - F_{3i}) \tag{4.12}$$

The above equations can be written in matrix form as

$$\Theta = (1 - F) \times Q \tag{4.13}$$

which is identical to Formula 4.4. Therefore, during the first step of the execution

$$\Theta = \begin{bmatrix} 0 & 1 - {}^{1}\!/_{e} & 1 - {}^{1}\!/_{e^2} \\ 1 - {}^{1}\!/_{e} & 0 & 1 - {}^{1}\!/_{e} \\ 1 - {}^{1}\!/_{e} & 1 - {}^{1}\!/_{e^2} & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 - {}^{1}\!/_{e} - {}^{1}\!/_{e^2} \\ 2 - {}^{2}\!/_{e} \\ 2 - {}^{1}\!/_{e} - {}^{1}\!/_{e^2} \end{bmatrix} \tag{4.14}$$

Hence, the first node that will be added in the move vector will be node 2. Node 2 is the one that minimizes the "entropy" of the network or the sum of vector Q. Node 2 is also an intuitive choice; it is the only vertex that influences the other 2 nodes in one step and also constitutes the best single-node move in this graph. The algorithm will then adjust Q according to Formula 4.5 as

$$Q = (1 - F_x)^{\mathrm{T}} \cdot Q = \begin{bmatrix} 1 - {}^{1}\!/_{e} \\ 0 \\ 1 - {}^{1}\!/_{e} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 - {}^{1}\!/_{e} \\ 0 \\ 1 - {}^{1}\!/_{e} \end{bmatrix} \tag{4.15}$$

Execution of the second step is identical and is omitted. The second node that will be picked is ID 3.

4.2.3 Strategy Principles

During the Q adjustment, nodes that are close to the node that was picked are drastically reduced so they are unlikely to be picked at a later stage since their neighborhood should already have low quota. Furthermore, nodes with more central role tend to have shorter distances from other nodes so the corresponding row of F is usually high, which decreases the corresponding row of Θ, making them better candidates. In a nutshell, centrality is achieved with F and spreading is achieved with the Q adjustment process.

4.3 Generalized Greedy Algorithm

Although this greedy algorithm is consistent with the strategy principles (Section 3.4), it has difficulties making selections for the first step(s) of the calculation. This is because the first selections (mainly the very first where the vector Q hasn't been adjusted) tend to be the most central vertices, which –as we will observe in Chapter 5– is not always the most optimal. In order to overcome this difficulty, a new parameter called *depth* is introduced, such that

$$0 \leq depth \leq m \tag{4.16}$$

where m is the number of allowed nodes per move. When $depth = 0$, the generalized greedy algorithm degenerates into the greedy algorithm of Section 4.2 while on $depth = m$ the algorithms behaves like the brute-force method. This method is illustrated on Diagram 4.4.

The principle of this algorithm is to manually enumerate all the possible sets of $depth$ amount of moves and let the normal greedy algorithm complete the sets with the remaining $(m - depth)$ moves. The best move from those sets is the one that achieves the minimum $sum(Q)$. A value of $depth = 1$ was used in the experiments in Chapter 5.

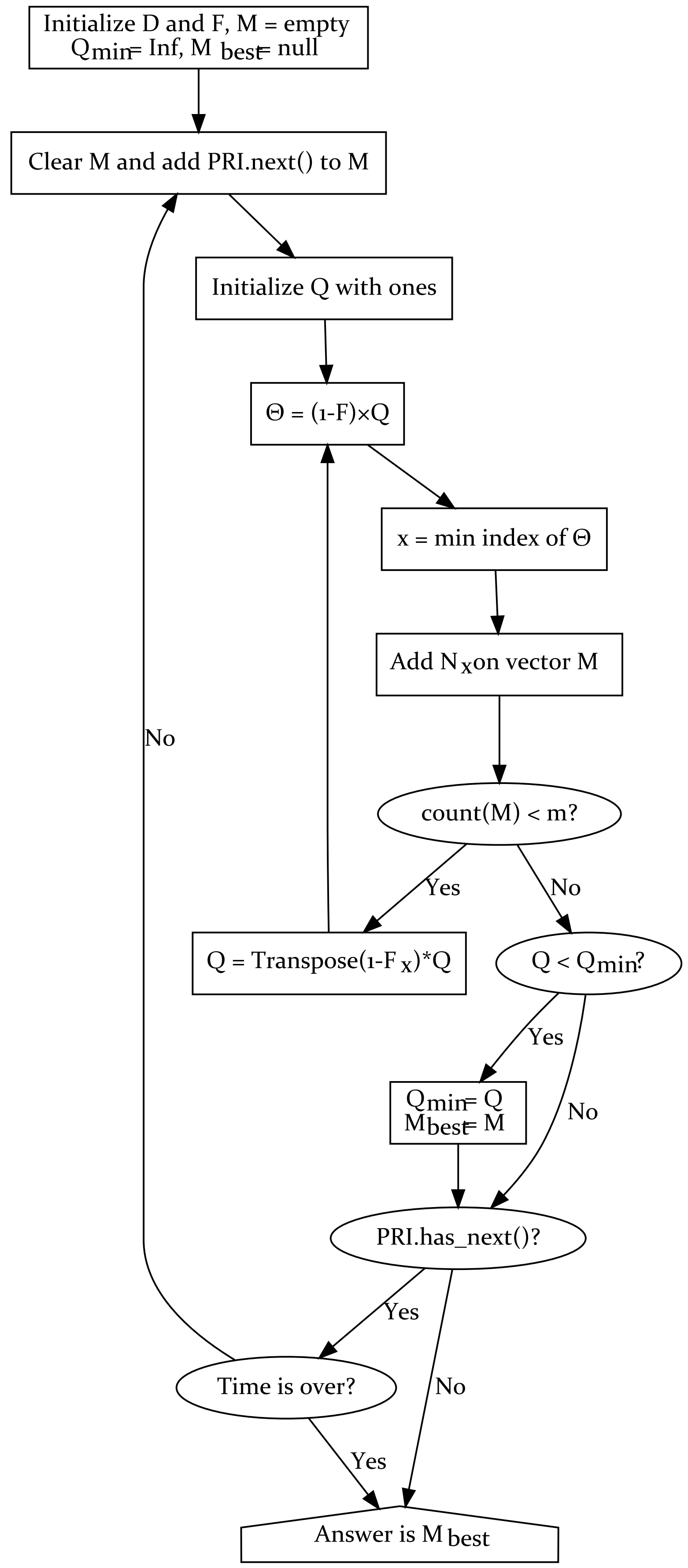

Figure 4.4: Diagram of the generalized greedy algorithm. *PRI* denotes the iterator which manually enumerates the sets of *depth* amount of moves.

Chapter 5

Experimental Results

In this chapter, all sets of tests that were performed will be presented in the form of tables and will be examined. The experiments are separated based on the graph type (all undirected): path graph, cycle graph, two wheel graph, scale-free graph and scale-free cluster graph. For each type of graph, several tests were conducted and overall conclusions were drawn.

The experiments were performed in a modern PC with Intel® Core™ i7-4770K CPU and 32GB of RAM running Ubuntu 13.04. The software tools[1] were developed in Java 7 using IntelliJ IDEA 13 Community Edition.

Since in many problem instances we are not aware of the optimal strategy, we evaluate our algorithms by comparing them with each other. Hence, the tests were done in a round-robin tournament-like environment and four players took place:

Random player – An agent who will pick the maximum number of nodes allowed at random.

Max PageRank player – An agent who will pick the maximum number of nodes allowed based on their PageRank value (damping factor 0.15). Vertices with the same PageRank centrality are ordered based on their ID.

Random brute-force search player – An agent implementing the brute-force algorithm mentioned in Section 4.1.

Greedy player – An agent implementing the greedy algorithm mentioned in Section 4.3. The *depth* parameter used on all tests was set to 1.

The parameters used in the tests are:

Graph – This is a set of parameters related to the construction of the graph, like number of vertices and other variables. Details are given later for each graph individually.

[1]`https://github.com/gstamatelat/social-influence`

Maximum move count – The maximum number of vertices that an agent can directly influence. Players in every experiment exhausted this limit. This measure will be mentioned as m.

Maximum execution time – This parameter affects only the brute-force player and the greedy one. When this limit is reached, the player will immediately return the best candidate move that it has up to that point. The values used for this parameter are (in seconds) 1, 15 and 30 and each triplet inside the cells on the tables below represent these execution times.

A total of 10 rounds were executed for each set of parameters while each round consisted of 6 games (since there are 4 participants). A win awards 1 point and a draw awards 0.5 points so total points sum to 60 for each test and each set of parameters. For every game a new graph is initialized; this practically affects the random-generated graphs.

Furthermore, for each graph and parameter set, a sample move suggested by the brute-force and greedy player will be noted for instructional purposes on separate tables. These moves were suggested using a 10-second execution time limit and are written in the form

$$[(N_1 = W_1), (N_2 = W_2), \ldots, (N_m = W_m)].$$

The better move between those two is highlighted in bold.

5.1 Path Graph

A path graph[2] is a particularly simple example of a tree, namely a tree with two or more vertices that is not branched at all, that is, contains only vertices of degree 2 and 1. A path graph with n vertices will have $n-1$ edges and a diameter of $n-1$. Path graphs do not have a DeGroot convergence, meaning that, given the adjacency matrix T, $\lim_{t\to\infty} T^t$ is undefined since it alternates between two values. Let's take for example the adjacency matrix for a path graph with $n = 4$:

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ {}^1/_2 & 0 & {}^1/_2 & 0 \\ 0 & {}^1/_2 & 0 & {}^1/_2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.1}$$

then

$$T^k = \begin{bmatrix} {}^1/_3 & 0 & {}^2/_3 & 0 \\ 0 & {}^2/_3 & 0 & {}^1/_3 \\ {}^1/_3 & 0 & {}^2/_3 & 0 \\ 0 & {}^2/_3 & 0 & {}^1/_3 \end{bmatrix} \tag{5.2}$$

[2] http://en.wikipedia.org/wiki/Path_graph

with k even and $k \longrightarrow \infty$. Also

$$T^{k+1} = \begin{bmatrix} 0 & ^2/_3 & 0 & ^1/_3 \\ ^1/_3 & 0 & ^2/_3 & 0 \\ 0 & ^2/_3 & 0 & ^1/_3 \\ ^1/_3 & 0 & ^2/_3 & 0 \end{bmatrix} \tag{5.3}$$

so the iterative DeGroot process will not converge. This is also because all cycles in this graph are of length that are multiple of 2 (see Section 2.2.2). The eigenvector centrality of the graph can be obtained by solving the equation $s\,T = s$:

$$s = factor \cdot \begin{bmatrix} 1 & 2 & 2 & 1 \end{bmatrix} \tag{5.4}$$

So, the (non-normalized) eigenvector centrality of this graph is $\begin{bmatrix} 1 & 2 & 2 & 1 \end{bmatrix}$. It is easy to reproduce the above method for larger path graphs:

$$\text{Eigenvector Centrality of Path Graph} = \begin{bmatrix} 1 & 2 & 2 & \dots & 2 & 2 & 1 \end{bmatrix} \tag{5.5}$$

PageRank on a path graph with n vertices will be lowest on 1 and n and highest on 2 and $n-1$. For example, the PageRank (damping factor 0.15) vector of a path graph with 6 vertices is

$$\text{PageRank of Path/6} = \begin{bmatrix} 0.658 & 1.196 & 1.145 & 1.145 & 1.196 & 0.658 \end{bmatrix} \tag{5.6}$$

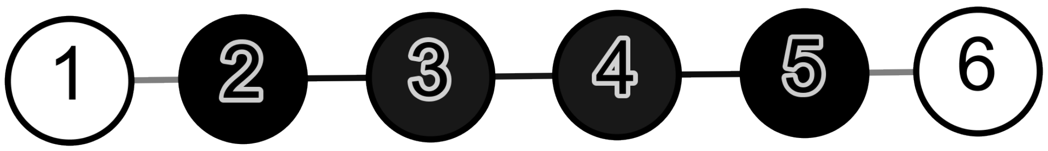

FIGURE 5.1: A path graph with 6 vertices colored by their PageRank value.

A path graph is only characterized by the number of nodes n it contains. $n = \{5, 11, 25\}$ was used in this set of tests.

(n, m)	Random	Max PageRank	Brute-Force	Greedy
(5,1)	4.5 – 5.0 – 6.5	7.5 – 6.0 – 5.0	**24.0** – 24.5 – 24.0	**24.0** – **24.5** – **24.5**
(5,2)	2.5 – 0.5 – 2.5	**19.5** – **20.0** – 19.0	18.5 – 19.5 – 19.0	**19.5** – **20.0** – **19.5**
(11,3)	4.0 – 6.0 – 7.0	6.0 – 4.0 – 3.0	**28.0** – **30.0** – **29.0**	22.0 – 20.0 – 21.0
(25,3)	9.0 – 9.0 – 9.0	1.0 – 1.0 – 1.0	23.0 – **26.0** – **29.0**	**27.0** – 24.0 – 21.0
(51,5)	14.0 – 11.0 – 11.0	0.0 – 0.0 – 0.0	16.0 – 19.0 – 19.0	**30.0** – **30.0** – **30.0**

TABLE 5.1: Path graph results

(n, m)	Brute-Force	Greedy
(5,1)	[3=1.0]	[3=1.0]
(5,2)	[2=1.0, 4=1.0]	[2=1.0, 4=1.0]
(11,3)	**[2=0.83, 6=1.11, 9=1.06]**	[3=1.0, 6=1.0, 9=1.0]
(25,3)	**[5=1.07, 13=0.74, 21=1.19]**	[6=1.0, 13=1.0, 20=1.0]
(51,5)	[3=0.9, 15=1.41, 25=0.63, 36=0.83, 46=1.23]	**[7=1.0, 16=1.0, 26=1.0, 35=1.0, 44=1.0]**

TABLE 5.2: Path graph sample moves

Overall, according to Table 5.1, the brute-force and the greedy players perform better. However, there are a few things one can note:

1. The brute-force will generally score higher on higher execution time limits.
2. The greedy algorithm scales very well with the increase of move nodes count. This is due to the complexity of brute-force, which slows down considerably on higher move node count.
3. Max PageRank performance peaks on $m = 2$. This is because on a path graph with 5 nodes, the 2 nodes with highest centrality are 2 and 4, which happens to be the best action.
4. The random player's strength increases as the graph gets larger and even outperforms the PageRank player. This (surprising) result can be attributed to the uniform distribution of the *random* function. Given enough space, the random player will make sure (up to a degree) that the random nodes are spread evenly throughout the path graph while the PageRank player picks will always tend to be near the edge (but not directly the edge). Spreading is more important in this sparse graph than centrality.

5.2 Cycle Graph

A cycle graph[3] is a graph that consists of a single cycle, or in other words, some number of vertices connected in a closed chain. A cycle graph with n vertices has n edges and every vertex has degree 2. A cycle graph with even number of vertices, like the path graph, does not converge. All vertices have the same eigenvector centrality and the same PageRank due to the graph's symmetry.

A cycle graph is also only characterized by the number of nodes n it contains. We used 4, 10, 24, 50 as n for our tests. On the cycle graph, spreading is the only factor that affects the optimal move since no vertex is more central than any other.

[3]http://en.wikipedia.org/wiki/Cycle_graph

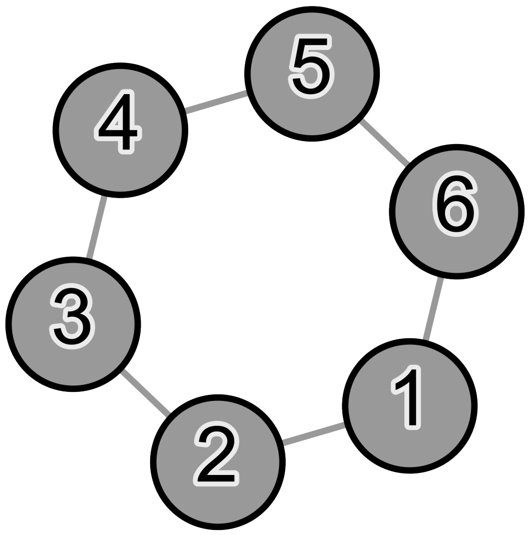

FIGURE 5.2: A cycle graph with 6 vertices colored by their PageRank value.

(n, m)	Random	Max PageRank	Brute-Force	Greedy
(4,1)	15.0 – 15.0 – 15.0	15.0 – 15.0 – 15.0	15.0 – 15.0 – 15.0	15.0 – 15.0 – 15.0
(4,2)	10.0 – 10.0 – 10.0	3.0 – 3.5 – 2.5	**24.0** – 22.5 – 24.0	23.0 – **24.0** – **23.5**
(10,3)	13.0 – 10.5 – 13.0	0.0 – 0.5 – 0.0	**30.0** – **29.0** – **30.0**	17.0 – 20.0 – 17.0
(24,3)	12.0 – 11.5 – 12.0	0.0 – 0.0 – 0.0	**30.0** – **30.0** – **30.0**	18.0 – 18.5 – 18.0
(50,5)	13.0 – 11.0 –10.0	0.0 – 0.0 – 0.0	17.0 – 22.0 – **25.0**	**30.0** – **27.0** – **25.0**

TABLE 5.3: Cycle graph results

(n, m)	Brute-Force	Greedy
(4,1)	[4=1.0]	[1=1.0]
(4,2)	[1=1.0, 3=1.0]	[1=1.0, 3=1.0]
(10,3)	**[3=0.97, 6=0.99, 10=1.04]**	[3=1.0, 6=1.0, 8=1.0]
(24,3)	**[2=0.94, 10=1.03, 18=1.03]**	[1=1.0, 7=1.0, 19=1.0]
(50,5)	[5=0.63, 16=0.68, 28=1.11, 38=1.76, 49=0.82]	**[8=1.0, 21=1.0, 33=1.0, 40=1.0, 46=1.0]**

TABLE 5.4: Cycle graph sample moves

Some conclusions deriving from Table 5.3 are the following:

1. On the $(4, 1)$ test, all players performed equally well. This, of course, is due to the fact that all single-vertex moves are identical since they all lead to symmetric graphs.
2. The PageRank player scores very low, especially on larger graphs. This is because all vertices have the same centrality and PageRank player will select them in order rather than randomly. Since the move nodes are cramped, the score is very low.
3. The greedy player is gaining a significant advantage over the brute-force one on the large graph. The score difference, however, is reduced in higher execution times because the

brute-force has a computational advantage.

4. The random player's performance is particularly good on almost all tests. This, like in the path graph but even stronger here, is due to the uniform distribution of the *random* function.

5.3 Two-Wheels Graph

The two-wheels graph is a graph which consists of two wheels bridged to a common vertex. This type of network can have n vertices, where n odd and $n \geq 7$, but for our tests we only used $n = 11$. This graph was used in order to demonstrate the importance of spreading the move nodes throughout the network over picking vertices with some high centrality measure. The PageRank (damping factor 0.15) in the two-wheels graph is

$$\text{PageRank/.15 of Two-Wheels} \approx \begin{cases} 1.554 & \text{for node 1} \\ 1.331 & \text{for nodes 2 and 3} \\ 0.857 & \text{for nodes 4, 6, 9, 10} \\ 0.839 & \text{for nodes 5, 7, 8, 11} \end{cases} \tag{5.7}$$

The eigenvector centrality in the two-wheels graph is

$$\text{Eigenvector centrality of Two-Wheels} = \begin{cases} 1 + {}^{13}/_{20} = 1.65 & \text{for node 1} \\ 1 + {}^{3}/_{8} = 1.375 & \text{for nodes 2 and 3} \\ {}^{33}/_{40} = 0.825 & \text{for the rest} \end{cases} \tag{5.8}$$

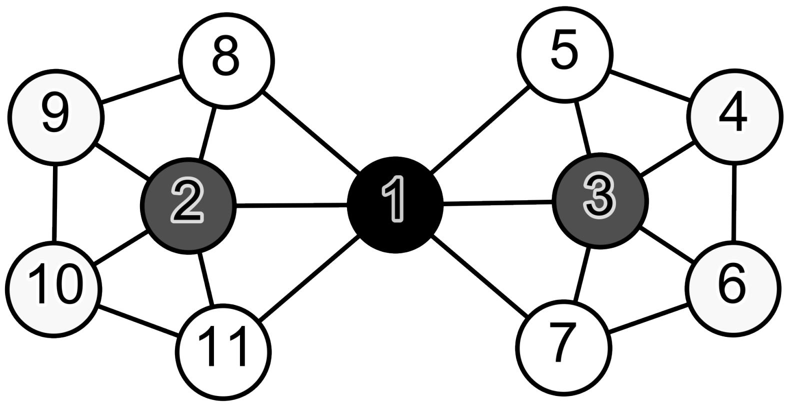

FIGURE 5.3: The two-wheels graph with vertices colored by their PageRank centrality value.

Since we only use 11 nodes for the two-wheels graph, the only parameter is m, the maximum number of vertices of a move.

(m)	Random	Max PageRank	Brute-Force	Greedy
(1)	1.0 – 2.0 – 2.5	**20.0** – **19.5** – **19.5**	19.5 – 19.0 – 19.0	19.5 – **19.5** – 19.0
(2)	0.5 – 0.0 – 1.0	9.5 – 10.0 – 9.0	20.5 – 24.0 – **25.0**	**29.5** – **26.0** – **25.0**
(3)	0.0 – 0.0 – 0.0	**25.0** – 19.0 – 20.0	10.0 – **23.0** – **22.0**	**25.0** – 18.0 – 18.0
(4)	1.0 – 2.0 – 2.0	14.0 – 13.0 – 14.0	**30.0** – **29.0** – **30.0**	15.0 – 16.0 – 14.0

Table 5.5: Two-Wheels graph results

(m)	Brute-Force	Greedy
(1)	[1=1.0]	[1=1.0]
(2)	[2=1.0, 3=1.0]	[2=1.0, 3=1.0]
(3)	[1=0.98, 2=0.98, 3=1.05]	**[1=1.0, 2=1.0, 3=1.0]**
(4)	**[1=1.12, 2=1.34, 3=0.98, 4=0.56]**	[1=1.0, 2=1.0, 3=1.0, 10=1.0]

Table 5.6: Two-Wheels graph sample moves

Some observations based on Table 5.5 are:

1. The random player has very poor performance on all tests. This comes to no surprise; careful planning is needed for a player to succeed on this graph due to its manipulated geometry.
2. On $m = 1$ all players (except random) perform equally well because all will suggest node 1, which is the best move.
3. The PageRank player fails on $m = 2$ and this is because it suggests $(1, 2)$ while the optimal move is $(2, 3)$. This is a typical example of a sparse graph that the top-rank vertex should not be included in the move.
4. On $m = 4$, the player based on the greedy algorithm scores poorly. This is partly due to the asymmetry of the graph. Moves with weighted nodes suggested by the brute-force player are generally stronger than moves with uniform weights (all nodes unary weight). The greedy player suggests $(1, 2, 3, 6)$ on this graph which is exactly the same as the suggestion of the max PageRank player.

5.4 Scale-Free Graph

A scale-free graph[4] is a graph whose degree distribution follows a power law, at least asymptotically. A few examples of networks claimed to be scale-free include:

1. Social networks, including collaboration networks. Two examples that have been studied extensively are the collaboration of movie actors in films and the co-authorship by mathematicians of papers.

[4] http://en.wikipedia.org/wiki/Scale-free_network

2. Many kinds of computer networks, including the Internet and the web graph of the World Wide Web.
3. Some financial networks such as interbank payment networks. [4, 17]
4. Protein-protein interaction networks.
5. Semantic networks. [18]
6. Airline networks.

Several scale-free models and generative methods exist, however, we use the Barabási-Albert [2] algorithm to generate scale-free graphs. According to this algorithm:

1. The network begins with an initial connected network of k_0 nodes.
2. New nodes are added to the network one at a time. Each new node is connected to $k \leq k_0$ existing nodes with a probability that is proportional to the number of links that the existing nodes already have.

Heavily linked nodes ("hubs") tend to quickly accumulate even more links, while nodes with only a few links are unlikely to be chosen as the destination for a new link. The new nodes have a "preference" to attach themselves to the already heavily linked nodes. Due to the algorithm itself, nodes with low ID tend to have higher centrality since they are added earlier on the network. This can be illustrated on Figure 5.4.

Experiments for this graph are given in the form (n, k_0, m), where n is the number of nodes in the graph, k_0 is the initial clique and m the number of move nodes.

(n, k_0, m)	Random	Max PageRank	Brute-Force	Greedy
(10,2,1)	1.0 – 1.0 – 0.5	19.0 – 19.5 – 20.0	**21.0** – **21.5** – **20.5**	19.0 – 18.0 – 19.0
(20,2,2)	0.0 – 0.0 – 0.0	24.0 – 17.0 – 15.0	11.0 – **27.0** – **30.0**	**25.0** – 16.0 – 15.0
(50,2,5)	10.0 – 0.0 – 0.0	**28.5** – **27.0** – **27.0**	1.0 – 16.0 – 14.0	20.5 – 17.0 – 19.0
(50,5,5)	7.0 – 0.0 – 0.0	**29.0** – **28.0** – **28.0**	3.0 – 15.0 – 16.0	21.0 – 17.0 – 16.0

TABLE 5.7: Scale-Free graph results

(n, k_0, m)	Brute-Force	Greedy
(10,2,1)	[2=1.0]	[2=1.0]
(20,2,2)	[1=1.0, 2=1.0]	[1=1.0, 2=1.0]
(50,2,5)	**[1=2.19, 3=1.3, 12=1.03, 22=0.05, 27=0.43]**	[1=1.0, 3=1.0, 12=1.0,] 16=1.0, 39=1.0]
(50,5,5)	[3=1.13, 4=1.7, 7=0.99, 15=0.63, 26=0.55]	**[2=1.0, 4=1.0, 5=1.0, 15=1.0, 45=1.0]**

TABLE 5.8: Scale-Free graph sample moves

Based on the results in Table 5.7 the following can be observed:

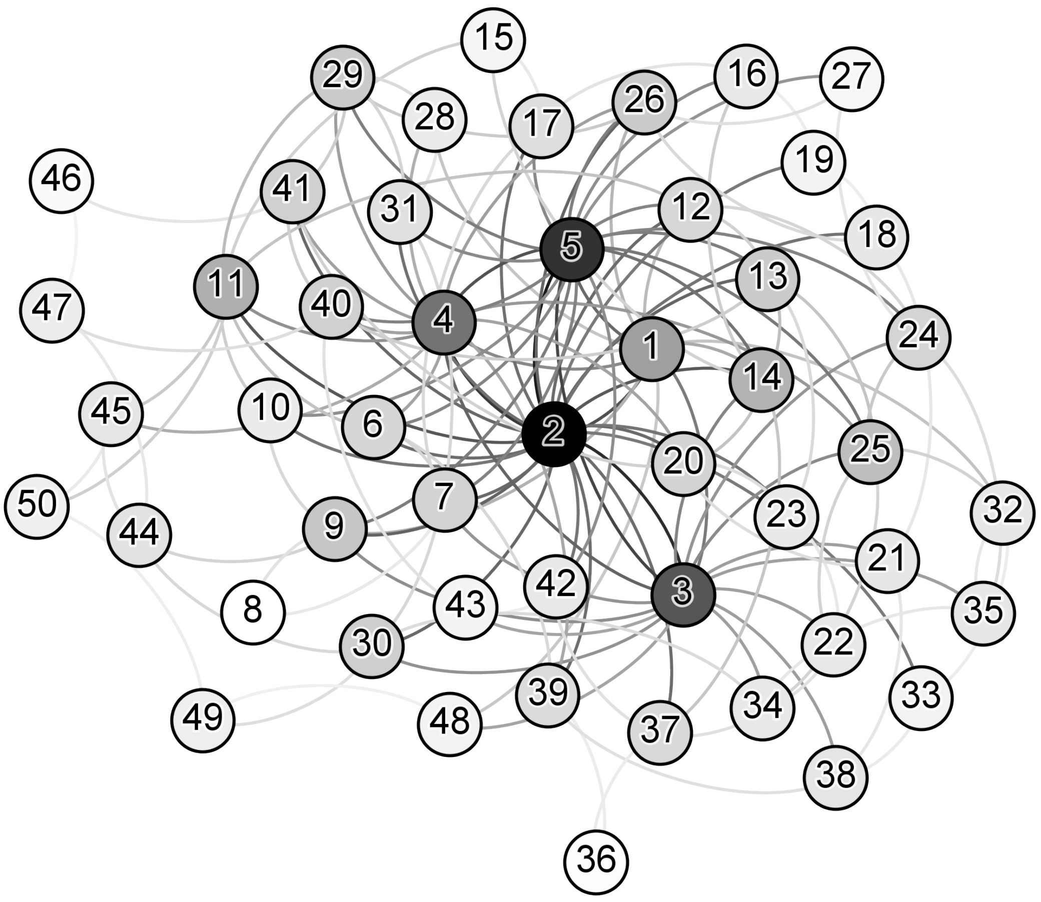

FIGURE 5.4: A scale-free graph with 50 vertices colored by their PageRank value. The seed used for this graph is 5785. The diameter of this graph is 4.

1. On $m = 5$, the random player can achieve a reasonable score using the uniform distribution mentioned earlier. On higher execution times, however, this performance is diminished by the brute-force player.

2. The PageRank player performs particularly well, especially on larger graphs. This is justified by the importance of selecting nodes with higher centrality values on dense graphs. It is worth noting that the graph illustrated on Figure 5.4 has a diameter of 4, which is the same as the two-wheels graph.

5.5 Scale-Free Cluster Graph

This scale-free cluster graph is composed of several scale-free sub-graphs bridged together in pairs. The bridged points are random vertices. The scale-free cluster graph is a combination of a cycle graph and a scale-free one.

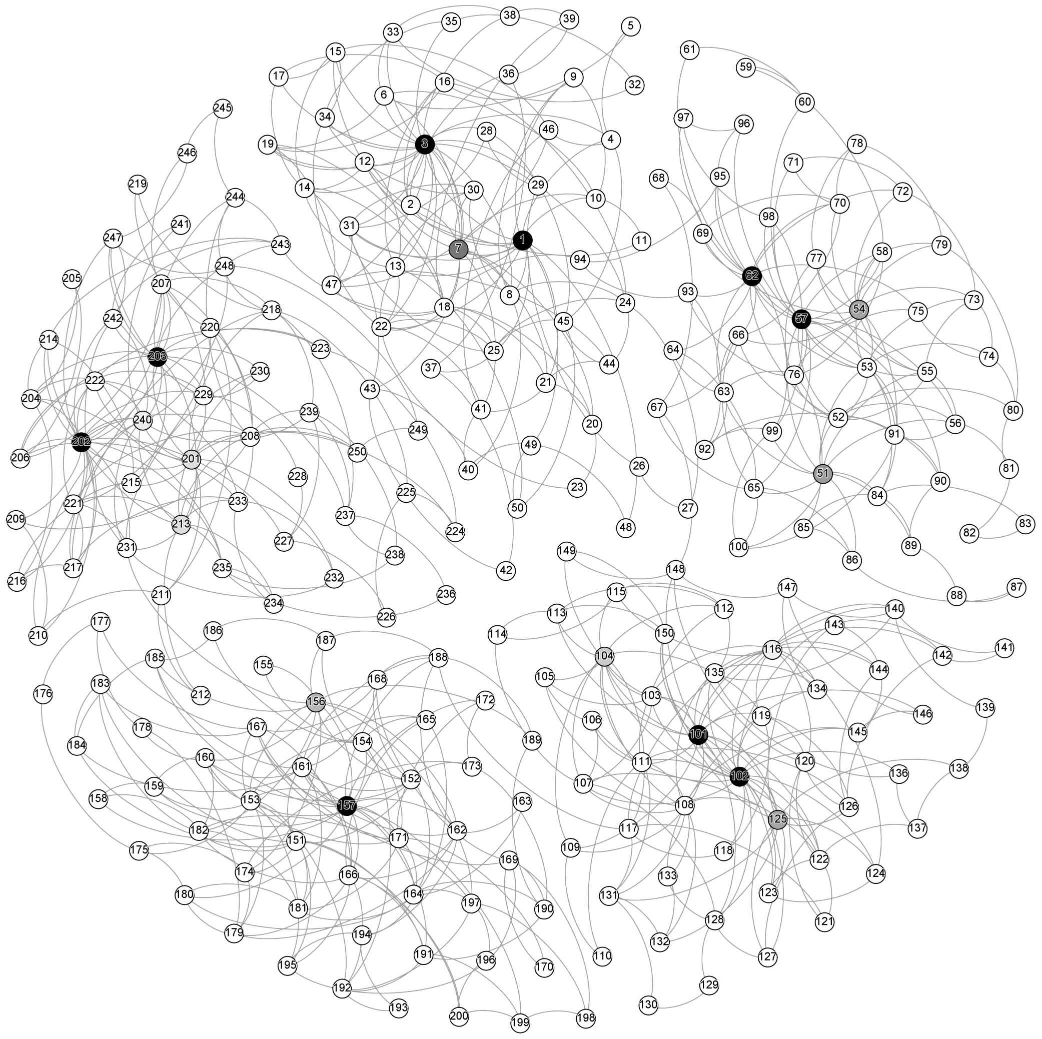

FIGURE 5.5: A scale-free cluster graph with 5 subgraphs of 50 vertices each colored by their PageRank value. The seed used for this graph is 310398. Its diameter is 13.

Experiments for this graph are given in the form (n, k_0, c, m), where n is the number of nodes in each sub-graph, k_0 is the initial clique, c the number of clusters and m the number of move nodes.

Observations for scale-free cluster graph based on Table 5.9:

1. The greedy player is the clear winner on low execution time limits and made the "best guess at first sight" on all tests.

2. The brute-force player performs poorly even on high execution times. This is especially true on high volumes of m because the complexity of the algorithm increases exponentially in relation to m.

(n, k_0, c, m)	Random	Max PageRank	Brute-Force	Greedy
(10,2,2,1)	8.5 – 2.0 – 3.0	18.0 – 13.5 – 12.5	7.5 – **25.0** – **25.5**	**26.0** – 19.5 – 19.0
(10,2,2,2)	2.0 – 3.0 – 2.0	19.5 – 15.0 – 12.5	16.0 – **22.5** – **28.0**	**22.5** – 19.5 – 17.5
(10,2,2,4)	0.0 – 4.0 – 2.0	20.5 – 12.5 – 20.5	17.0 – **22.0** – 21.0	**22.5** – 21.5 – **21.5**
(25,2,5,1)	3.5 – 2.5 – 2.0	20.5 – 11.5 – 12.0	12.5 – **27.0** – **29.5**	**23.5** – 19.0 – 16.5
(25,2,5,5)	2.0 – 0.0 – 0.0	21.5 – 14.5 – 13.0	8.0 – 16.0 – 17.0	**28.5** – **29.5** – **30.0**
(25,2,5,10)	1.0 – 0.0 – 0.0	23.0 – **25.0** – 23.0	9.0 – 10.0 – 11.0	**27.0** – **25.0** – **26.0**

TABLE 5.9: Scale-Free Cluster graph results

(n, k_0, c, m)	Brute-Force	Greedy
(10,2,2,1)	[3=1.0]	[3=1.0]
(10,2,2,2)	**[1=0.99, 13=1.01]**	[2=1.0, 13=1.0]
(10,2,2,4)	[4=1.09, 9=0.91, 11=0.91, 14=1.1]	**[5=1.0, 9=1.0, 11=1.0, 14=1.0]**
(25,2,5,1)	**[36=1.0]**	[76=1.0]
(25,2,5,5)	[13=1.37, 27=0.49, 32=0.35 52=1.46, 102=1.34]	**[7=1.0, 27=1.0, 52=1.0, 80=1.0, 101=1.0]**
(25,2,5,10)	[1=1.06, 14=1.34, 32=1.1, 42=0.78, 54=0.9, 59=0.72, 79=1.39, 100=0.42, 104=1.16, 118=1.13]	**[2=1.0, 11=1.0, 27=1.0, 28=1.0, 51=1.0, 52=1.0, 77=1.0, 101=1.0, 102=1.0, 104=1.0]**

TABLE 5.10: Scale-Free Cluster graph sample moves

Chapter 6

Discussion

In this thesis, we defined and examined a non-cooperative, zero-sum game between two firms on a social network. We investigated possible equilibria and stated primitive directions towards a successful strategy. We also analyzed and compared several algorithms implementing player strategies: a completely random player, a player that always selects the highest PageRank-value vertices, a player implementing a brute-force search technique and, finally, a player that implements a greedy algorithm which uses a heuristic to suggest an action.

6.1 Strategy Principles

The most important goal of this work was to obtain some general insight on the game and a basic understanding on its mechanics. This was accomplished with careful observations using the random brute-force search algorithm mentioned in Section 4.1 and the findings were three strategy principles mentioned in Section 3.4. Of course, these principles do not apply to all types of graphs or to any circumstances but rather provide a rough guide to what appears to be a decent technique of maximizing the utility function.

These principles suggest that a player should select all the nodes m that are allowed and also not to select vertices that are close to each other but rather span over the entire graph. The player should also try to select nodes with a relatively high centrality measure without sacrificing the first principle.

6.2 Algorithm Conclusions

Random – The random player only performs well on graphs where spreading is the major factor that determines the optimal strategy. These graphs include the path and cycle graph, especially the latter, and are usually very sparse. This behavior is attributed to the uniform distribution of the *random* function, which is also the reason of the huge fluctuation among graphs that characterizes the random player.

Max PageRank – On the other hand, the max PageRank player achieved higher score on denser graphs like the scale-free one and the scale-free cluster. These graphs have low diameter compared to their size and spreading the move isn't very meaningful; centrality is the formula of success in this case. Like the random player, max PageRank has also a large amount of fluctuation, ranging from best (on scale-free) to worst (on the path).

Brute-Force – A general remark on the random brute-force search algorithm is that although it suffers from prohibitively high complexity, it constantly reports the progress and any new moves that it finds. Hence, in practice, it will quickly find a relatively decent move without the need of exhausting all the combinations of moves. This is due to the fact that is gets progressively harder to achieve a better move once a decent one has been established.

The random brute-force search player performed best on low values of n and m, but also scored firmly on all types of graphs where other players failed. This algorithm is dominant on high enough values of execution time and was used extensively to provide general insights, strategy principles and game mechanics understanding.

Greedy – The greedy algorithm achieved a very consistent performance throughout all tests but its major strength proved to be the complicated scale-free cluster graph. Furthermore, it provided the majority of the best "moves at first sight" (1 second execution).

6.3 Future Work

Primarily, in order to better understand the mechanics of the game we need to further examine and experiment with the convergence for problem instances which do not have pure Nash equilibria. This perspective of the problem is rather vague for now.

Furthermore, it is suspected that the speed of convergence and the effectiveness of a player action may be related; it is possible that a superior move can lead to faster convergence. This is something that should be investigated.

We also need to enrich the tests with more types of graphs and possibly incorporate parts of real Internet social networks, like Twitter. It would be very interesting to examine the behavior of our algorithms in such a network.

Finally, adding additional players and tweaking the existing ones could help further establish the validity of the strategy principles mentioned in this thesis. The greedy algorithm could quite possibly perform better with modifications on the distance matrix D (and therefore the matrix F) to better reflect the metric of influence or trust rather than distance. There are also thoughts about an algorithm based on the concept of force-directed graph drawing (see [13] and the references therein), which function by assigning forces between connected vertices and simulating the graph up to the equilibrium state. This idea can also be utilized by assigning a negative electrical charge on each vertex of the graph and adding m positively charged entities in the graph. These points will tend to be attracted by high concentrations of graph vertices (centrality) but will repel each other (spreading).

Appendix A

Figure and Diagram Sources

All figures in this thesis were created using the open-source software Gephi [3] using "ForceAtlas 2" layout algorithm.

Diagrams were created using the DOT graph description language[1] and rendered with the software Graphviz[2]. Below is the source code for the diagrams.

Diagram of the brute-force algorithm – Figure 4.1

```
digraph G {
 bgcolor = transparent;
 graph [fontname = "Constantia"];
 node [fontname = "Constantia"];
 edge [fontname = "Constantia"];

 get_initial [shape=box, label="C = GetRandomMove()"];
 get_random [shape=box, label="R = GetRandomMove()"];
 is_better [label="Is R better than C?"];
 assignment [shape=box, label="C = R"];
 time_over [label="Is time over?"];
 answer [shape=house, label="Answer is C"];

 get_initial -> get_random;
 get_random -> is_better;
 is_better -> time_over [label="No"];
 is_better -> assignment [label="Yes"];
 time_over -> get_random [label="No"];
 assignment -> time_over;
 time_over -> answer [label="Yes"];
}
```

Diagram of the greedy algorithm – Figure 4.2

```
digraph G {
 bgcolor = transparent;
```

[1] http://en.wikipedia.org/wiki/DOT_(graph_description_language)
[2] http://www.graphviz.org/

```
 graph [fontname = "Constantia"];
 node [fontname = "Constantia"];
 edge [fontname = "Constantia"];

 initialize [shape=box, label="Matrices D, F and vectors\nQ and M are initialized"];
 get_qi [shape=box, label⊠=" = (1-F)×Q"];
 get_x [shape=box, label="x = min index of ⊠"];
 add_nx [shape=box, label=<Add N<SUB>x</SUB> on vector M>];
 get_q [shape=box, label=<Q = Transpose(1-F<SUB>x</SUB>)*Q>];
 finished [label="count(M) < m?"];
 answer [shape=house, label="Answer is M"];

 initialize -> get_qi;
 get_qi -> get_x;
 get_x -> add_nx;
 add_nx -> finished;
 finished -> answer [label="No"];
 finished -> get_q [label="Yes"];
 get_q -> get_qi;
}
```

Diagram of the generalized greedy algorithm – Figure 4.4

```
digraph G {
 bgcolor = transparent;
 graph [fontname = "Constantia"];
 node [fontname = "Constantia"];
 edge [fontname = "Constantia"];

 init [shape=box, label=<Initialize D and F, M = empty<BR/>
   Q<SUB>min</SUB> = Inf, M<SUB>best</SUB> = null>];
 initialize [shape=box, label="Initialize Q with ones"];
 get_qi [shape=box, label⊠=" = (1-F)×Q"];
 get_x [shape=box, label="x = min index of ⊠"];
 add_nx [shape=box, label=<Add N<SUB>x</SUB> on vector M>];
 get_q [shape=box, label=<Q = Transpose(1-F<SUB>x</SUB>)*Q>];
 finished [label="count(M) < m?"];
 q_less_min [label=<Q &lt; Q<SUB>min</SUB>?>];
 pri_has_next [label="PRI.has_next()?"];
 assign_q_less [shape=box, label=<Q<SUB>min</SUB> = Q<BR/>M<SUB>best</SUB> = M>];
 answer [shape=house, label=<Answer is M<SUB>best</SUB>>];
 time_over [label="Time is over?"];
 clear_m [shape=box, label="Clear M and add PRI.next() to M"];

 initialize -> get_qi;
 get_qi -> get_x;
 get_x -> add_nx;
 add_nx -> finished;
 finished -> get_q [label="Yes"];
 get_q -> get_qi;
 finished -> q_less_min [label="No"];
 q_less_min -> pri_has_next [label="No"];
 q_less_min -> assign_q_less [label="Yes"];
 assign_q_less -> pri_has_next;
 pri_has_next -> answer [label="No"];
 pri_has_next -> time_over [label="Yes"];
 time_over -> answer [label="Yes"];
 time_over -> clear_m [label="No"];
 clear_m -> initialize;
 init -> clear_m;
}
```

Bibliography

[1] W. Allport Gordon. The nature of prejudice, 1954.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286 (5439):509–512, 1999.

[3] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009. URL http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

[4] G. De Masi, G. Iori, and G. Caldarelli. Fitness model for the italian interbank money market. *Phys. Rev. E*, 74:066112, Dec 2006. doi: 10.1103/PhysRevE.74.066112. URL http://link.aps.org/doi/10.1103/PhysRevE.74.066112.

[5] M. H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69 (345):118–121, 1974.

[6] P. Dubey, R. Garg, and B. De Meyer. Competing for customers in a social network: The quasi-linear case. In *Internet and Network Economics*, pages 162–173. Springer, 2006.

[7] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[8] J. Ghaderi and R. Srikant. Opinion dynamics in social networks: a local interaction game with stubborn agents. In *American Control Conference (ACC), 2013*, pages 1982–1987. IEEE, 2013.

[9] B. Golub and M. O. Jackson. Naive learning in social networks and the wisdom of crowds. *American Economic Journal: Microeconomics*, pages 112–149, 2010.

[10] S. Goyal and M. Kearns. Competitive contagion in networks. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 759–774. ACM, 2012.

[11] M. O. Jackson. *Social and economic networks*. Princeton University Press, 2010.

[12] J. Kleinberg. *Networks, Crowds, and Markets*. Cambridge University Press, 2010.

[13] S. G. Kobourov. Spring embedders and force directed graph drawing algorithms. *arXiv preprint arXiv:1201.3011*, 2012.

[14] M. Newman. *Networks: an introduction.* Oxford University Press, 2010.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[16] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[17] K. Soramäki, M. L. Bech, J. Arnold, R. J. Glass, and W. E. Beyeler. The topology of interbank payment flows. *Physica A: Statistical Mechanics and its Applications*, 379(1):317–333, 2007.

[18] M. Steyvers and J. B. Tenenbaum. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive science*, 29(1):41–78, 2005.

[19] E. Yildiz, D. Acemoglu, A. Ozdaglar, A. Saberi, and A. Scaglione. Discrete opinion dynamics with stubborn agents. *SSRN eLibrary*, 2011.