ALGORITHMIC APPLICATIONS TO SOCIAL NETWORKS, SECRETARY PROBLEMS AND DIFFERENTIAL PRIVACY

BY Mangesh Gupte

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Michael Saks

and approved by

_____________________

_____________________

_____________________

_____________________

_____________________

New Brunswick, New Jersey

October, 2011
ABSTRACT OF THE DISSERTATION

Algorithmic Applications to Social Networks, Secretary Problems and Differential Privacy

by Mangesh Gupte
Dissertation Director: Michael Saks

This dissertation is a study in the applications of algorithmic techniques to four specific problems in the domains of social networks, online algorithms and differential privacy.

We start by looking at the structure of directed networks. In [Gupte, Shankar, Li, Muthukrishnan, and Iftode, 2011] we define a measure of hierarchy in directed online social networks, and using primal-dual techniques, present an efficient algorithm to compute this measure. Our experiments on different online social networks show how hierarchy emerges as we increase the size of the network.

We study the dynamics of information propagation in online social networks in [Gupte, Hajiaghayi, Han, Iftode, Shankar, and Ursu, 2009]. For a suitable random graph model of a social network, we prove that news propagation follows a threshold phenomenon, hence, high-quality information provably spreads throughout the network assuming users are greedy. Starting from a sample of the Twitter graph, we show through simulations that the threshold phenomenon is exhibited by both the greedy and courteous user models.

In chapter 4, we turn our attention from social networks to online algorithms. The
The specific problem we tackle is to select a large independent set from a general independence set systems when elements arrive online and we are in the random permutation model. Using a random sampling oracle, we give an algorithm which outputs a set of size \( \Omega \left( \frac{s}{\log n \log \log n} \right) \), where \( s \) is size of the largest set. This gives a \( O(\log n \log \log n) \) approximation algorithm, which matches the lower bound within \( \log \log n \) factors.

The last problem we discuss is a scheme for the private release of aggregate information [Gupte and Sundararajan, 2010]. Such a scheme must resolve the trade-off between utility to information consumers and privacy of the database participants. Differential privacy [Dwork, McSherry, Nissim, and Smith, 2006] is a well-established definition of privacy. In this work, we present a universal treatment of utility based on the standard minimax rule from decision theory. Under the assumption of rational behavior, we show that for every fixed count query, a certain geometric mechanism is universally optimal for all minimax information consumers. Additionally, our solution makes it possible to release query results at multiple levels of privacy in a collusion-resistant manner.
Acknowledgements

This thesis has been the culmination of work done over the last six years and so a lot of people contributed to making it happen. A lot of the early work did not make it into the thesis, however, it did shape the kind of problems I later worked on and that are present in the thesis. I would like to thank everyone who helped me throughout this journey.

Most of all, I would like to thank my adviser, Mike Saks, who has been a support throughout. His patience with me as I wandered around trying out different problems and areas has been remarkable. I have particularly enjoyed exploring problems with him. I have also had the pleasure of working with S. Muthukrishnan, Rebecca Wright and Liviu Iftode. The breadth of topics I could work on which contributed greatly to my enjoyment in the program. I would also like to thank Michael Littman, Tina Eliassi-Rad and Martin Farach-Colton whom I worked with at different stages of the Ph.D.

I would like to thank my committee members Mike, Muthu, Rebecca and Vahab for taking time read and give helpful comments on the thesis. The quality improved significantly and it would have been much poorer without their efforts.

Concentrating on work and fun was possible because of the great work by the staff in my department. I would especially like to thank our graduate secretary Carol DiFrancesco who took care of the all the day to day administrative issues I faced and Aneta Unrat who took the payroll, insurance and other issues out of my way.

Working in the Computer Science department has been a pleasure due to my colleagues working in theory. The engaging discussions with Rajat, Fengming, Nikos,
Devendra, Darakshan, Alex, Arzoo and Alantha were a constant source of problems and new directions. My office mates Pavel and Edi made staying in office fun.

Over the years, I ended up having a long list great of housemates: Sankar, Datla, Vardhan, Siddharth, Pinakin, Rajesh and Shivesh. A notable feature of my stay were the long lunches and the variety of discussion topics in the Busch Campus Center, where I spent countless hours, working and otherwise. A lot of friends kept me company there and I would especially like to mention to mention Rajat who was the prime organizer of lunches in BCC and Praveen, Hari, Gayatree, Rekha and Devendra who made the lunches and discussions fun. Work cannot happen without an equal dose of fun and the most fun I have had was during the numerous parties and game nights. Thank you Shruti, Shivesh, Gayatree, Pallavi, Vikas, Madhur, Shiva and Rajesh for showing how easy it is to keep awake for the whole night when you have something worthwhile to do! Sagar, Mukund, Anubhav and Sangeeta have always been around since my pre-Rutgers days to discuss course problems, startup dreams, life expectations, jobs, internships and everything else under the sun.

Family is what kept me going throughout my stay here. Shweta has been supporting me (both emotionally and financially!) for many years now without too much of a grumble :-) Meeting her and being with her here at Rutgers has been the highlight of the Ph.D. Aai, Baba and Poornima tai all been pushing me to always do better and to complete faster. Their belief in me and their support has been invaluable. Of course, my twin nephews Aditya and Pranav, who arrived during the Ph.D. adding spice to life, making it more hectic and interesting and also very entertaining.
Dedication

To my wife Shweta
and to my parents.

To friends in the noisy booths of BCC
and the delicious lunches of Faculty Dining.

To long nights of getting high
on board games
and in parties
and especially on enchanting new ideas.

To hard problems
and technical proofs
and clumsy errors
and aimless exploration.

To the six years of work
and enjoyment
and fun
and growth.

To new beginnings ...
# Table of Contents

Abstract ................................................................. ii  
Acknowledgements ......................................................... iv  
Dedication ................................................................. vi  
List of Tables ........................................................... x  
List of Figures .......................................................... xi  

1. Introduction ......................................................... 1  

2. Hierarchy in Social Networks ................................. 7  
   2.1. Introduction ..................................................... 7  
   2.2. Hierarchy in Directed Social Networks ....................... 9  
       2.2.1. Examples .................................................. 11  
   2.3. Efficiently Measuring Hierarchy ............................. 13  
       2.3.1. Algorithm ............................................... 13  
   2.4. Proofs ........................................................ 18  
   2.5. Experiments .................................................... 22  
       2.5.1. Validation of the Hierarchy Measure ............... 24  
       2.5.2. Importance of Edge Direction ...................... 25  
       2.5.3. Hierarchy in Online Social Networks vs Random Graphs 27  
       2.5.4. Effect of Scaling on Social Hierarchy .......... 29  
   2.6. Related Work .................................................. 34
2.7. Conclusions and Future Directions ........................................ 35

3. **News Propagation in Social Networks** .................................. 36
   
   3.1. Introduction ............................................................... 36
   
   3.2. Strategic User Model ................................................... 38
   
   3.3. Random Graph Models of Social Networks .......................... 40
       
       3.3.1. Connectivity of stochastic Kronecker graphs ............... 42
   
   3.4. Analysis of a Model of Strategic User Behavior ................. 43
       
       3.4.1. The Connectedness of $G[R]$ .................................. 45
   
   3.5. Simulation Results .................................................... 50
   
   3.6. Conclusion and Future Work ......................................... 52

4. **Secretary Problems on Independence Set Systems** ..................... 57
   
   4.1. Introduction ............................................................... 57
   
   Online Algorithms .......................................................... 57
   
   4.1.1. The Random Permutation Model .................................. 58
       
   4.1.2. Independence Set Systems ....................................... 59
       
   4.1.3. The Secretary Problem on Independence Set Systems ........ 60
       
   4.1.4. Related Work .......................................................... 61
       
   4.1.5. Bipartite Graph representation of Independence Set Systems . 63
   
   4.2. Oracles and Problem Variants ....................................... 63
   
   4.3. Algorithm ................................................................. 65
   
   4.4. Weighted Elements ..................................................... 70
   
   4.5. Future direction using the Oracle $O_2$ ............................ 71
       
       4.5.1. Approximating the offline oracle .............................. 71
       
       4.5.2. A labeling game ................................................ 73
   
   4.6. Conclusion ............................................................... 76
5. Differential Privacy .................................................. 79

5.1. Introduction ....................................................... 79

5.2. Model and Results ............................................... 82
   5.2.1. Privacy Mechanisms and Differential Privacy ............ 82
   5.2.2. Oblivious Mechanisms ...................................... 84
   5.2.3. Minimax Information Consumers ............................. 85
   5.2.4. Interactions of Information Consumers with Mechanisms . . 86
       Motivation ......................................................... 86
       Feasible Interactions ........................................... 86
       Optimal Interactions ........................................... 87
   5.2.5. Optimal Mechanism for a Single Known Information Consumer . 88
   5.2.6. Optimal Mechanism for Multiple Unknown Information Consumers 89
   5.2.7. Comparison with Bayesian Information Consumers .......... 92
   5.2.8. Related Work ............................................... 93

5.3. Characterizing Mechanisms Derivable from the Geometric Mechanism . 94

5.4. Applications of the Characterization ............................ 98
   5.4.1. Information-Consumers at Different Privacy Levels .......... 98
   5.4.2. Universal Utility Maximizing Mechanisms .................. 100

5.5. There always exists an Optimal Mechanism that is Oblivious .......... 105

5.6. A Differentially Private Mechanism that is not derivable from the Geometric Mechanism ........................................... 108

5.7. Conclusion ...................................................... 108
List of Tables

5.1. The optimal mechanism for consumer $c$ characterized by loss-function

\[ l(i, r) = |i - r| \] 
and side-information $S = \{0, 1, 2, 3\}$. $n = 3, \alpha = 1/4$. . . 90

5.2. The Range Restricted Geometric Mechanism . . . . . . . . . . . . . . . . 95
## List of Figures

2.1. Graphs with perfect hierarchy. \( h(G) = 1 \) for each of these graphs. Nodes labels indicate levels. All edges have agony 0.  

2.2. Graphs with some hierarchy. All unlabeled edges have agony 0.  

2.3. Graphs with no hierarchy. \( h(G) = 0 \) for each of these graphs. All edges have agony 1.  

2.4. Correlation of hierarchy with popular metrics.  

2.5. Correlation with Wikipedia notability score.  

2.6. Hierarchy in the College Football network.  

2.7. Hierarchy in random graphs.  

2.8. Hierarchy in social network among famous people.  

2.9. Effect of network size on hierarchy.  

2.10. Distribution of ranks among nodes.  

2.11. Distribution of ranks among nodes.  

2.12. Effect of directed edges.  

3.1. Coverage of \textit{greedy} and \textit{courteous} strategies over the crawled graph. \( q \) is the probability with which each user likes the link. \( t \) is the threshold for the greedy strategy. \( c \) is the threshold for the courteous strategy.  

4.1. Modeling knapsack constraints as an independence set system  

4.2. Independence set systems as a bipartite graph.  

4.3. Bucketing the independent sets.
5.1. The probability distribution on outputs given by the Geometric

Mechanism for $\alpha = 0.2$ and query result 5. . . . . . . . . . . . . . . 83
Chapter 1

Introduction

The computer science landscape of changed drastically over the last few years. Online social networks exploded on the scene in the second half of the last decade, touching not just areas within Computer Science itself, but also the lives of people outside the field. Sites like Facebook and Twitter have become household names. Facebook has more than 750 million active users, half of whom log on any given day. Online social networks have become an increasingly popular medium for sharing information such as links, news and multimedia among users. For example, during the recent Iran elections, traditional news media acknowledged the power and influence of social networks such as Twitter. In this scenario, understanding the structure of social networks and the dynamics of sharing in networks is a fundamental challenge. Graph theory gives us the necessary tools to embark on this. We analyze the structure of directed social networks in chapter 2 and the dynamics of the process of news propagation in social networks in 3.

The advent of social networks has been fueled by the immense amount of data that people share in these networks. With the rising popularity of social networks and in an age of information sharing, privacy concerns have become paramount. Initial attempts at privacy were based on syntactic definitions like like $k$-anonymity [Sweeney, 2002]. The break in privacy of seemingly anonymous datasets like the one released by Netflix as part of their programming challenge [Narayanan and Shmatikov, 2008] have shown the need for refined definitions of privacy. Differential privacy [Dwork, McSherry, Nissim, and Smith, 2006] gives a semantic definition of privacy and places privacy on
sound theoretical foundations. Mechanisms guarantee differential privacy by perturbing results – they add random noise to the query result, and guarantee protection against all attackers, whatever their side-information or intent. We discuss a universal privacy preserving mechanism in chapter 5.

Finally, we study online algorithms in chapter 4. We look at the problem of selecting a large independent set of elements in the secretary setting.

Underlying the study of all these problems is a common set of algorithmic techniques like linear programming and duality, the theory of random graphs and concentration bounds. These techniques help us analyze and understand modern phenomena like online social networks. This dissertation is a study in the applications of algorithmic and combinatorial techniques to different domains. We shall look at four specific problems in the domains of social networking, online algorithms and differential privacy. The modus operandi is to look at real world problems, construct simplified models of the phenomenon under consideration, analyze the simplified models and apply the findings back to the problem we started out with.

We start with the work in [Gupte, Shankar, Li, Muthukrishnan, and Iftode, 2011] that looks at the structure of directed networks. Social hierarchy and stratification among humans is a well studied concept in sociology. The popularity of online social networks presents an opportunity to study social hierarchy for different types of networks and at different scales. We adopt the premise that people form connections in a social network based on the perceived social hierarchy; as a result, the edge directions in directed social networks can be leveraged to infer hierarchy. We define a measure of hierarchy in a directed online social network, and present an efficient algorithm to compute this measure. The main challenge is to define a measure that captures the intuitive meaning of hierarchy while also being efficiently computable. We use primal-dual linear programming techniques to come up with a polynomial time combinatorial algorithm to compute this measure. We validate that our measure corresponds with
hierarchy as observed by people, by using ground truth including the Wikipedia notability score. We then use the measure to study hierarchy in several directed online social networks including Twitter, Delicious, YouTube, Flickr, LiveJournal, and curated lists of several categories of people based on different occupations, and different organizations. Our experiments on different online social networks show how hierarchy emerges as we increase the size of the network. This is in contrast to random graphs, where the hierarchy decreases as the network size increases. Further, we show that the degree of stratification in a network increases very slowly as we increase the size of the graph.

In [Gupte, Hajiaghayi, Han, Iftode, Shankar, and Ursu, 2009], we look at the dynamics of information propagation in online social networks. We argue that users in social networks are strategic in how they post and propagate information. We propose two models of user behavior — the greedy and the courteous model — and study information propagation both analytically and through simulations. We model social networks as random graphs and use properties of random graphs to prove how news propagates over social networks. We model news using a single parameter $q$ that denotes the quality. This parameter intuitively corresponds to the fraction of the population that finds the particular news piece interesting. We also give two plausible models of user behavior: greedy and courteous. For suitable random graph models of social networks, we prove that news propagation follows a threshold phenomenon, hence, high-quality information provably spreads throughout the network assuming users are greedy. Starting from a sample of the Twitter graph, we show through simulations that the threshold phenomenon is exhibited by both the greedy and courteous user models.

The next problem we tackle deals with online algorithms. The specific problem we tackle is to select a large independent set from a general independence set systems when elements arrive online. In the worst case (adversarial) setting, we cannot hope to select more than one element. But in the random permutation model, also called the secretary setting, we give an algorithm which outputs a set of size $\frac{s}{\log n}$ with
probability $O\left(\frac{1}{\log \log n}\right)$, where $s$ is size of the largest set and $n$ is the total number of elements in the set system. This gives a $O(n \log \log n)$ approximation algorithm, which matches the lower bound within $O\left((\log \log n)^2\right)$ factor.

Last, we turn our attention to the problem of privacy. We need a scheme to release aggregates of private information that are also useful [Gupte and Sundararajan, 2010]. A scheme that publishes aggregate information about sensitive data must resolve the trade-off between utility to information consumers and privacy of the database participants. Differential privacy [Dwork, McSherry, Nissim, and Smith, 2006] is a well-established definition of privacy—this is a universal guarantee against all attackers, whatever their side-information or intent. In this work, we present a universal treatment of utility based on the standard minimax rule from decision theory. In our model, information consumers are minimax (risk-averse) agents, each possessing some side-information about the query, and each endowed with a loss-function which models their tolerance to inaccuracies. Further, information consumers are rational in the sense that they actively combine information from the mechanism with their side-information in a way that minimizes their loss. Under the assumption of rational behavior, we show that for every fixed count query, a certain geometric mechanism is universally optimal for all minimax information consumers. Additionally, our solution makes it possible to release query results at multiple levels of privacy in a collusion-resistant manner. We use linear algebraic and linear programming techniques to solve this problem.
Social Networks
Chapter 2

Hierarchy in Social Networks

2.1 Introduction

Social stratification refers to the hierarchical arrangement of individuals in a society into divisions based on various factors such as power, wealth, knowledge and importance. Stratification existed among humans since the very beginning of human society and continues to exist in modern society. In some settings, such as within an organization, the hierarchy is well known, whereas in other settings, such as conferences and meetings between a group of people, the hierarchy is implicit but discernible.

The popularity of online social networks has created an opportunity to study sociological phenomenon at a scale that were earlier unfathomable. Phenomenon such as small diameter in social networks [Travers and Milgram, 1969] and strength of weak ties [Granovetter, 1973] have been revisited in light of the large data now available about people and their connections [Albert, Jeong, and Barabási, 1990, Watts and Strogatz, 1998, Barabási, 2005]. Online social networks present an opportunity to study how social hierarchy emerges.

Scientists have observed dominance hierarchies within primates. Schjelderup-Ebbe [1975] showed a pecking order among hens where each hen is aware of its place among the hierarchy and there have been various papers that investigate the importance of such a hierarchy [Frank and Goyal, 2003, Fama and French, 2002]. However, data from experimental studies indicates that the dominance graph contains cycles and hence,  

\footnote{http://www.answers.com/topic/social-stratification-1}
does not represent true hierarchy. There has been a lot of work on extracting a chain given this dominance graph [de Vries, 1995, 1998, Appleby, 1983].

Stratification is manifested among humans in the form of a social hierarchy, where people higher up in the hierarchy have higher social status than people lower in the hierarchy. With the wide adoption of online social networks, we can observe the network and can leverage the links between nodes to infer social hierarchy. Most of the popular online social networks today, such as Twitter, Flickr, YouTube, Delicious and LiveJournal contain directed edges. Our central premise is that there is a global social rank that every person enjoys, and that individuals are aware of their rank as well as the rank of people they connect to.

Given a social graph, we cannot directly observe the ranks of people in the network, we can only observe the links. We premise that the existence of a link indicates a social rank recommendation; a link $u \rightarrow v$ ($u$ is a follower of $v$) indicates a social recommendation of $v$ from $u$. If there is no reverse link from $v$ to $u$, it might indicate that $v$ is higher up in the hierarchy than $u$. We assume that in social networks, when people connect to other people who are lower in the hierarchy, this causes them social agony. To infer the ranks of the nodes in the network, we find the best possible ranking, i.e. the ranking that gives the least social agony.

In this chapter, we define a measure that indicates how close the given graph is to a true hierarchy. We also give a polynomial time algorithm to evaluate this measure on general directed graphs and to find ranks of nodes in the network that achieve this measure.

We use our algorithm to measure hierarchy in different online social networks, including Twitter, Delicious, Flickr, YouTube, LiveJournal, and curated lists of people based on categories like different occupations, and different organizations.

We experimentally find, using a college football dataset, that the edge directions

\footnote{Facebook is an exception with undirected edges.}
encodes hierarchy information. The social strata of people in online social networks, measured using our metric, shows strong correlation with human-observed ground truth such as Wikipedia notability, as well as other well-known metrics such as page rank and friend-follower ratio. Our experiments show that hierarchy emerges as the size of an online social network grows. This is in contrast to random graphs, where the hierarchy decreases as the network size increases. Finally, we show that hierarchy in online social networks does not grow indefinitely; instead, there are a small number of levels (strata) that users are assigned to and this number does not grow significantly as the size of the network increases.

The key contributions that we see in this chapter are:

1. We define a measure of hierarchy for general directed networks.

2. We give a polynomial time algorithm to find the largest hierarchy in a directed network.

3. We show how hierarchy emerges as the size of the networks increases for different online social networks.

4. We show that, as we increase the size of the graph in our experiments, the degree of stratification in a network does not increase significantly.

2.2 Hierarchy in Directed Social Networks

One of the most popular ways to organize various positions within an organization is as a tree. A general definition of hierarchy is a (strict) partially ordered set. This definition includes chains (Figure 2.1a) and trees (Figure 2.1b) as special cases. We can view a partially ordered set as a graph, where each element of the set is a node and the partial ordering \((u > v)\) gives an edge from \(u\) to \(v\). The fact that the graph represents a partial order implies that the graph is a Directed Acyclic Graph (DAG). From now
on, we use DAGs as examples of perfect hierarchy. Figure 2.1c shows an example of a DAG.

Let us define a measure of hierarchy for directed graphs that might contain cycles. Consider a network \( G = (V, E) \) where each node \( v \) has a rank \( r(v) \). Formally, the rank is a function \( r : V \rightarrow \mathbb{N} \) that gives an integer score to each vertex of the graph. Different vertices can have the same score.

In social networks, where nodes are aware of their ranks, we expect that higher rank nodes are less likely to connect to lower rank nodes. Hence, directed edges that go from lower rank nodes to higher rank nodes are more prevalent than edges that go in the other direction. In particular, if \( r(u) < r(v) \) the edge \( u \rightarrow v \) is expected and does not cause any agony to \( u \). However, if \( r(u) \geq r(v) \), then edge \( u \rightarrow v \) causes agony to the user \( u \) and the amount of agony depends on the difference between their ranks. We shall assume that the agony caused to \( u \) by each such reverse edge is equal to \( r(u) - r(v) + 1 \).\(^3\)\(^4\) Hence, the agony to \( u \) caused by edge \((u, v)\) relative to a ranking \( r \) is \( \max(r(u) - r(v) + 1, 0) \).

We define the agony in the network relative to the ranking \( r \) as the sum of the agony on each edge:

\[
A(G, r) = \sum_{(u,v) \in E} \max(r(u) - r(v) + 1, 0)
\]

We defined agony in terms of a ranking, but in online social networks, we can only observe the graph \( G \) and cannot observe the rankings. Hence, we need to infer the rankings from the graph itself. Since nodes typically minimize their agony, we shall find a ranking \( r \) that minimizes the total agony in the graph. We define the agony of

\(^3\)Note that \( r(u) - r(v) \) does not work, since it gives rise to trivial solutions like \( r = 1 \) for all nodes. The +1 effectively penalizes such degenerate solutions. Using any positive constant threshold \( c \) other than 1 does not change the optimal ranking, the minimum agony value gets scaled by a factor of \( c \).

\(^4\)An interesting direction for future work is to investigate a different measure of agony, in particular, a non-linear function like \( \log(r(u) - r(v) + 1) \).
$G$ as the minimum agony over all possible rankings $r$:

$$A(G) = \min_{r \in \text{Rankings}} \left( \sum_{(u,v) \in E} \max(r(u) - r(v) + 1, 0) \right)$$

For any graph $G$, $A(G)$ is upper bounded by $m$, the number of edges in $G$ (we prove this in Section 2.3.1 Equation 2.1). This motivates our definition of hierarchy in a graph:

**Definition 2.1 (Hierarchy).** The hierarchy $h(G)$ in a directed graph $G$ is defined as

$$h(G) = 1 - \frac{1}{m} A(G)$$

$$= \max_{r \in \text{Rankings}} \left( 1 - \frac{1}{m} \sum_{(u,v) \in \text{edges}} \max(r(u) - r(v) + 1, 0) \right)$$

For any graph $G$, the hierarchy $h(G)$ lies in $[0,1]$. This follows from the fact that $A(G)$ lies in $[0,m]$. (Equation 2.1). To gain some intuition into this definition of hierarchy, we shall look at some example graphs and their hierarchy.

### 2.2.1 Examples

![Graphs](image)

(a) A chain  
(b) A tree  
(c) A DAG

Figure 2.1: Graphs with perfect hierarchy. $h(G) = 1$ for each of these graphs. Nodes labels indicate levels. All edges have agony 0.

DAGs have perfect hierarchy. $h(G) = 1$ when $G$ is a DAG. This is achieved by setting $r(v) > r(u) + 1$ for each edge $(u,v)$ in the DAG. Figure 2.1 shows examples of
graphs with perfect hierarchy. Nodes are labeled with levels. For this assignment, note that the agony on each edge is 0.

Consider the graph in Figure 2.2a. The hierarchy of this graph is $1 - \frac{1}{6} \times 2 = \frac{2}{3}$. If instead of the edge $(r, l_1)$, the “deeper” edge $(r, l_2)$ is present, as shown in Figure 2.2b, then the hierarchy of the new graph becomes $1 - \frac{1}{6} \times 4 = \frac{1}{3}$. This illustrates how hierarchy changes in a very simple setting. We shall explore this more in Section 2.5.

Directed cycles have no hierarchy. $h(G) = 0$ when $G$ is a collection of edge disjoint
directed cycles. We prove in Section 2.3.1 that for any assignment of labels to nodes, the agony is at least $m$. Figure 2.3 shows examples of graphs with 0 hierarchy. If each node is labeled the same, say 1, this is achieved.

### 2.3 Efficiently Measuring Hierarchy

To find the hierarchy $h(G)$ for a given graph $G$, we need to search over all rankings and find the best one. Since the number of rankings $r$ is exponentially large, we need an efficient way to search among them. In Section 2.3.1, we present an efficient algorithm that given a directed graph $G$ as input, finds as output a ranking of the vertices of $G$ that gives the highest hierarchy for the input graph $G$.\(^5\)

#### 2.3.1 Algorithm

In this section, we describe an algorithm that finds the optimal hierarchy for a given directed graph $G = (V, E)$. For notational convenience, we shall denote $n = |V|$ and $m = |E|$. For a scoring function $r : V \rightarrow \mathbb{N}$, the hierarchy relative to $r$ is:

$$h(G, r) = 1 - \frac{1}{m} \sum_{(i, j) \in E} \max(r(i) - r(j) + 1, 0)$$

The task is to find an $r$ such that $h(G, r)$ is maximized over all scoring functions. But maximizing $h$ is the same as minimizing the total agony $A(G, r)$. We formulate minimizing agony as the following integer program:

\(^5\)This ranking may not be unique. In fact, if $G$ is a DAG, then any ordering that gives a topological sort of $G$ gives an optimal ranking.
\[
\begin{align*}
\min & \sum_{(i,j) \in E} x(i,j) \\
x(i, j) & \geq r(i) - r(j) + 1 \quad \forall (i,j) \in E \\
x(i, j) & \geq 0 \quad \forall (i,j) \in E \\
r(i) & \geq 0 \quad \forall i \in V \\
x(i,j), r(i) & \in \mathbb{Z}
\end{align*}
\]

We can interpret the variables in the linear program as follows: the \(x(i, j)\) are defined for each pair \((i,j) \in E\) and represents the agony on each edge. The variables \(r(i)\) give the rank of each node. We see that any feasible solution to this linear program gives a ranking and the value of the linear program measures the total agony. We now see a simple upper bound on the minimum value of the integer program. Consider the solution:

\[
r(i) = 0 : \forall i \in V \\
x(i, j) = 1 : \forall (i,j) \in E
\]

(2.1)

This is clearly feasible and the objective value for this is \(m\). This gives a simple upper bound of \(m\) on the objective value of the above integer program and hence on the maximum agony in any graph.

To get insight into this problem, we look at the linear relaxation of this integer program and then form the dual linear program. The dual is:

\[
\begin{align*}
\max & \sum_{(i,j) \in E} z(i,j) \\
z(i, j) & \leq 1 \quad \forall (i,j) \in E \\
\sum_{j \in V} z(k, j) & \leq \sum_{i \in V} z(i,k) \quad \forall k \in V \quad \text{(node-degree)} \\
z(i, j) & \geq 0 \quad \forall (i,j) \in E
\end{align*}
\]
We can strengthen the node-degree constraints without affecting the solution of the linear program by requiring equality, since if we sum over all $k$, we get:

$$\sum_{k \in V} \sum_{j \in V} z(k, j) \leq \sum_{k \in V} \sum_{i \in V} z(i, k)$$

Since both sides count the total number of edges in the graph, they are equal. Hence, equality must hold for each individual constraint as well. So, we can rewrite the node-degree condition as:

$$\sum_{j \in V} z(k, j) = \sum_{i \in V} z(i, k) \quad \forall k \in V$$  \hspace{1cm} \text{(node-degree)}

Suppose we restrict the dual variables to be 0 or 1 instead of in the range $[0, 1]$, then we can think of the variables $z(i, j)$ as being indicator variables for the edge $(i, j)$. The constraint $\sum_{j \in V} z(k, j) = \sum_{i \in V} z(i, k) \quad \forall k \in V$, then says that at each vertex $k$, the number of incoming edges must be equal to the number of outgoing edges. Hence, the dual program is finding the maximum (in terms of number of edges) Eulerian subgraph of the original graph.$^6$

The reinterpretation gives us insight into the primal solution. By weak duality, the value of the primal is lower bounded by the value of any feasible dual solution. Hence, the primal value cannot become smaller than the size of the maximum Eulerian subgraph.

If the original graph $G$ is Eulerian, this gives a lower bound of $m$. Equation 2.1 demonstrates a way to get $m$ as the primal solution. Hence, the optimal primal value for Eulerian graphs is in fact $m$. This proves the observation that, for graphs that are a collection of directed cycles, the agony is $m$ and hence, the hierarchy is 0.

We can directly solve the LP to get the best ranking when we do not restrict the rank of the node to be an integer. We shall prove that the linear program has an integral optimal solution. In fact, we give a combinatorial algorithm that finds the best ranking.

---

$^6$We say that a subgraph is Eulerian if the indegree of each vertex is equal to its outdegree. We do not impose the requirement that the subgraph be connected.
We first use Algorithm 1 to construct an integral solution to the dual. Algorithm 2 uses the dual solution to come up with an integral primal solution. We show that the primal and dual solutions have the same objective value which, by LP duality, proves that both are optimal.

Algorithm 1 constructs a maximum Eulerian subgraph of $G$. Theorem 2.2 proves the correctness of Algorithm 1; that the subgraph is Eulerian and also that it has the maximum number of edges among such subgraphs. We leave the proofs of Theorems 2.2 and 2.3 to Section 2.4.

**Theorem 2.2.** Let $H$ be the subgraph of $G$ that contains the reverse of all (and only those) edges labeled $+1$ by Algorithm 1. Then, for each vertex $v$ : $\text{indeg}_H(v) = \text{outdeg}_H(v)$. Also, for every subgraph $T$ of $G$ such that $\text{indeg}_T(v) = \text{outdeg}_T(v) : \forall v \in T$, the number of edges in $H$ is greater than the number of edges in $T$.

To find the optimal value of hierarchy in the graph $G$, we need to assign a score $r$ to the nodes and calculate the agony $x(i,j)$ value on each edge $(i,j)$. Algorithm 2 gives a labeling for each node, from the $\pm 1$ edge labels given by Algorithm 1. The input graph to Algorithm 2 is the one output by Algorithm 1.

Even though the graph output by Algorithm 1 has negative edges, it does not have any negative cycles and Lemma 2.7 proves that the algorithm terminates. Theorem 2.3 proves that the labels produced by this algorithm are optimal labels for the primal, and hence, produce the optimal hierarchy.

**Theorem 2.3.** The output $x,l$ of Algorithm 2 is a feasible solution to the primal. The output $z$ of Algorithm 1 is a feasible solution to the dual problem. Further,

$$\sum_{(u,v) \in E} x(u,v) = \sum_{(u,v) \in E} z(u,v)$$

Hence, $(x,l; z)$ give optimal solutions to the primal and dual linear programs respectively.
Algorithm 1: Finding a Maximum Eulerian Subgraph

Input: Graph $G = (V,E)$

Output:

1. A subgraph $H$ of $G$ such that $H$ is Eulerian and has the maximum number of edges.

2. A DAG such that $H \cup \text{DAG} = G$

Set the weight of each edge in $G$ as $w(u,v) \leftarrow -1$

Set $G_0 = G, H_0 = \phi$ and $s = 0$. We shall successively form graphs $G_0, \ldots, G_S$ and graphs $H_0, \ldots, H_S$

while $\exists$ a negative cycle $C$ in $G_s$ do

$G_{s+1} \leftarrow G_s$

for edge $(u,v) \in C$ do

$w(u,v) \leftarrow -w(u,v)$

Reverse the direction of the edge

$G_{s+1} \leftarrow G_{s+1} - (u,v) + (v,u)$

end

$H_{s+1} \leftarrow \text{reverse of all edges in } G_{s+1} \text{ labeled -1}$

Set $s \rightarrow s + 1$

end

DAG $\leftarrow \text{All edges labeled -1}$

$H \leftarrow H_S \ (H \text{ is Eulerian})$

$z(u,v) \leftarrow 0$, for edge $(u,v) \in \text{DAG}$

$z(u,v) \leftarrow 1$, for edge $(u,v) \in H$
Algorithm 2: Label the graph given as a decomposition of the Eulerian graph and a DAG

**Input:** A Graph $G = (V, H \cup \text{DAG})$ output by Algorithm 1. Edges in the Eulerian graph $H$ are labeled $+1$ and edges in $\text{DAG}$ are labeled $-1$.

**Output:** A labeling $l$ of all vertices of $G$, such that the agony measure on $G$ with the given labels: $A(G,l)$, is equal to the size of the Eulerian graph $H$.

Set label $l(v) \leftarrow 0$, for each vertex $v \in V$

while $\exists$ edge $(u,v)$ such that $l(v) < l(u) - w(u,v)$ do

\[ l(v) \leftarrow l(u) - w(u,v) \]

end

$x(u,v) \leftarrow 0$, for edge $(u,v) \in \text{DAG}$

$x(u,v) \leftarrow l(u) - l(v) + 1$, for edge $(u,v) \in H$

This shows that the value of the primal solution is equal to the value of a dual solution, which shows that both are optimal. We present the proof in Section 2.4.

### 2.4 Proofs

We shall now prove Theorem 2.2 and 2.3. We start with proving that Algorithm 1 produces a feasible dual solution.

**Lemma 2.4.** Let $H$ be the subgraph of $G$ that contains the reverse of all (and only those) edges labeled $+1$ by Algorithm 1. Then, for each vertex $v$ : $\text{indeg}_H(v) = \text{outdeg}_H(v)$

**Proof.** Let $H$ be the subgraph of $G$ consisting of all the $+1$ edges. We track the evolution of $H$ through steps $s = 0, \ldots, S$. Initially, $H_0$ is the empty graph. We establish the following loop invariants.

- All edges with label -1 belong to $G$. The reverse of all edges labeled +1 belong to $G$.
- $\forall v \in V, \forall s \in \{0, \ldots, S\} : \text{indeg}_{H_s}(v) = \text{outdeg}_{H_s}(v)$. 
These are true at the start when \( s = 0 \). If we prove these for each iteration of the loop, they will imply the lemma.

The first assertion is true, since we initialize the label all edges to \(-1\) and whenever we reverse an edge, we also change its sign.

We prove the second assertion by induction on \( s \). Suppose the assertion is true at some middle state \( H_s \). Algorithm 1 finds a directed cycle \( C \) in \( G_s \). Then \( H_{s+1} \) is formed from \( H_s \) by removing edges in \( C \) with label \(+1\) adding edges with label \(-1\) to \( H_{s+1} \).

For any vertex \( v \), the edges \( e_1, e_2 \) adjoining it in \( C \) can have any of the four \( \pm 1 \) label combinations. When they have labels \(+1,+1\), \( \text{indegree}_{H_{s+1}}(v) = \text{indegree}_{H_s}(v) - 1 \) and \( \text{outdegree}_{H_{s+1}}(v) = \text{outdegree}_{H_s}(v) - 1 \).

When they have labels \(-1,-1\), \( \text{indegree}_{H_{s+1}}(v) = \text{indegree}_{H_s}(v) + 1 \) and \( \text{outdegree}_{H_{s+1}}(v) = \text{outdegree}_{H_s}(v) + 1 \).

When the labels are \(-1,+1\), we remove edge \( e_2 \), which was pointing into \( v \) and add edge \( e_1 \), which now points into \( v \). Hence, \( \text{indegree}_{H_{s+1}}(v) = \text{indegree}_{H_s}(v) \) and \( \text{outdegree}_{H_{s+1}}(v) = \text{outdegree}_{H_s}(v) \). Similarly, if the labels were \(+1,-1\) then we remove edge \( e_2 \), which was pointing out of \( v \) in \( H \) and add edge \( e_1 \), which now points out of \( v \).

So, \( \text{indegree}_{H_{s+1}}(v) = \text{indegree}_{H_s}(v) \) and \( \text{outdegree}_{H_{s+1}}(v) = \text{outdegree}_{H_s}(v) \).

Since, by induction hypothesis \( \text{indegree}_{H_s}(v) = \text{outdegree}_{H_s}(v) \), we have shown that \( \text{indegree}_{H_{s+1}}(v) = \text{outdegree}_{H_{s+1}}(v) \). This proves the second assertion and completes the proof of the lemma.

\[ \square \]

**Lemma 2.5.** \( H \) is a maximum Eulerian subgraph.

**Proof.** Let \( T \) be another subgraph, such that number of edges of \( T \) is greater than number of edges of \( H \). Let \( \text{rev}(H) \) be the graph with edges of \( H \) reversed. Consider the graph \( P \) obtained by taking the disjoint union of edges of \( \text{rev}(H) \) and \( T \) and removing cycles of length two with one edge from \( H \) and the other from \( T \). Set the label of edges in \( \text{rev}(H \setminus T) \) to \( 1 \), and the label of edges in \( T \setminus H \) to \(-1\). The edges in \( T \cap H \) become
cycles of length two in \( \text{rev}(H) \cup T \) and are removed from \( P \). Observe that \( P \) occurs as a subgraph (along with the correct \( \pm 1 \) labels) of \( G_S \) at the termination of Algorithm 1.

\( P \) is Eulerian since both \( \text{rev}(H) \) and \( T \) are Eulerian and we only remove cycles from their disjoint union. Hence, we can construct a cycle cover of the edges of \( P \). But the total number of negative edges of \( P \) is greater than the number of positive edges. Hence, there exists a negative cycle in this cover. Since \( P \) is a subgraph of \( G \), this also implies that there exists a negative cycle in \( G \) at the end of the Algorithm 1, which is a contradiction.

**Lemma 2.6.** Algorithm 1 terminates in \( O(m^2n) \) time.

**Proof.** In each iteration of the loop, the number of edges with label +1 increases by at least 1. The total number of edges is upper bounded by \( m \). Hence, there are at most \( m \) iterations. Each iteration calculates a negative cycle detection algorithm, which can be done by Bellman-Ford and takes time \( O(mn) \) [Cormen, Leiserson, and Rivest, 2001, Section 24.1]. Hence, the total time is at most \( O(m^2n) \).

Hence, we have proved Theorem 2.2. Theorem 2.2 shows that Algorithm 1 calculates the optimal integral dual solution. We now prove properties of Algorithm 2. First we prove that Algorithm 2 terminates.

**Lemma 2.7.** If the input graph to Algorithm 2 does not contain negative cycles, then Algorithm 2 terminates.

**Proof.** All nodes have label 0 at the start of the algorithm. Consider the shortest paths between all pairs of vertices. Since there are no negative cycles, these are well defined. If the graph has no negative edges, then it cannot have negative cycles. Hence, let us consider the case when it does have negative edges. Let \( p \) be the minimum path length among all shortest paths. Since there exist negative edges, \( p \) will be negative. We claim that \(-p\) is an upper bound on the label that any vertex can get. If any vertex gets a
higher label, we can trace the set of edges that were used to get to that label, and these would give a shorter path (path with greater number of negative edges) than $p$, which is a contradiction.

The next lemma helps us prove Theorem 2.3.

**Lemma 2.8.** For each edge $(u, v) \in \text{DAG}$, $l(v) \geq l(u) + 1$. For each edge $(u, v) \in \text{the Eulerian subgraph } H$, $l(u) - l(v) + 1 \geq 0$

**Proof.** Suppose $(u, v) \in \text{DAG}$. Then, $w(u, v) = -1$. Hence, at the end of Algorithm 2, the condition $l(v) \geq l(u) - (-1)$ is satisfied. Similarly, for edge $(u, v)$ in $H$, $w(v, u) = 1$. Hence, at the end of Algorithm 2, the condition $l(u) \geq l(v) - 1$ is satisfied.

The above lemma shows setting the primal variables $x(u, v) = 0$ for an edge $(u, v)$ in the DAG, and $x(u, v) = l(u) - l(v) + 1 \geq 0$ for an edge $(u, v)$ in the Eulerian subgraph results in a feasible primal solution by Lemma 2.8.

**Theorem 2.3.** The output $x, l$ of Algorithm 2 is a feasible solution to the primal. The output $z$ of Algorithm 1 is a feasible solution to the dual problem. Further,

$$\sum_{(u,v) \in E} x(u,v) = \sum_{(u,v) \in E} z(u,v)$$

Hence, $(x, l; z)$ give optimal solutions to the primal and dual linear programs respectively.

**Proof.** Lemma 2.8 proves that $x, l$ is a feasible primal solution. Theorem 2.2 shows that $z$ is a feasible dual solution. Now, we show that the value of the primal solution is equal
to the value of a dual solution, which shows that both are optimal.

\[
\text{Value of the primal solution} = \sum_{(u,v) \in E} x(u,v)
\]

\[
= \sum_{(u,v) \in E} \max\{0, l(u) - l(v) + 1\}
\]

\[
= \sum_{(u,v) \in\text{DAG}} \max\{0, l(u) - l(v) + 1\} + \sum_{(u,v) \in\text{H}} \max\{0, l(u) - l(v) + 1\}
\]

\[
= 0 + \sum_{(u,v) \in\text{H}} l(u) - l(v) + 1 \quad \text{(By Lemmas 2.8)}
\]

\[
= \sum_{C \in\mathcal{C}} \sum_{(u,v) \in C} l(u) - l(v) + 1
\]

(\text{where } \mathcal{C} \text{ is some cycle cover of the Eulerian subgraph})

\[
= \sum_{C \in\mathcal{C}} |C| \left(\text{For any cycle } C, \sum_{(u,v) \in C} l(v) - l(u) + 1 = |C|\right)
\]

\[
= \text{number of edges in the Eulerian subgraph}
\]

\[
= \sum_{(u,v) \in\text{E}} z(i,j)
\]

\[
= \text{Value of the dual solution}
\]

This proves that \(x, l\) is an optimal primal solution and that \(z\) is an optimal dual solution. \(\square\)

This shows that the linear program has an integral optimal solution and that Algorithms 1 and 2 calculate the optimal solution to the integer program we started out with.

### 2.5 Experiments

In this section, we present the results of our experiments, which have the following goals:
• Validate that the notion of hierarchy we propose does correspond to real hierarchy based on ground truth.

• Validate that direction of edges does encode information about hierarchy.

• Compare how hierarchy emerges in online social graphs of different types of people, by using random graphs as baseline.

• Show how hierarchy emerges as the size of the social graph grows, for different online social networks.

Figure 2.4: Correlation of hierarchy with popular metrics.
2.5.1 Validation of the Hierarchy Measure

In this section, we want to validate that our measure of hierarchy corresponds with real hierarchy observed by humans. Towards this aim, we designed the following experiment. We collected a curated list of 961 journalists on Twitter and computed the hierarchy on the graph generated by these journalists among themselves. We also looked at how different measures of noteworthiness of node correlate with the level of that node in the optimal hierarchy. We now present the results.

The computed hierarchy measure is 0.38. This indicates that there is a medium hierarchy in this graph. There are seven levels (strata) that users are assigned to in the optimal hierarchy. A higher level indicates people who enjoy higher social status.

Wikipedia notability: To confirm that our computed hierarchy corresponds to a real hierarchy, we make use of Wikipedia to derive ground truth. Each node(journalist) is assigned a Wikipedia notability score, which is either No Entry (the person does not have an entry in Wikipedia), Small, Medium or Large (depending on the size of the Wikipedia entry). Figure 2.5 shows how our hierarchy measure compares with the ground truth obtained from Wikipedia. The figure shows that nodes with a low hierarchy level do not have a Wikipedia entry, and nodes higher up in the computed hierarchy are more likely to be noteworthy according to Wikipedia. This result lends credence to our measure of hierarchy.

Correlation with well known measures: To get more insight into the factors that contribute to a node’s hierarchy level, we measure the correlation of our computed hierarchy level for the journalists graph with well known measures of social networks: pagerank, friend-follower ratio, and Twitter list score.

Figure 2.4a plots the median page rank (along with the 10th and the 90th percentile value) for each hierarchy level. The figure shows that people with a high page rank tend to be higher up in the social hierarchy level computed by our measure.

Figure 2.4b plots the correlation of hierarchy level with the Twitter list score, which
corresponds to the number of user-generated Twitter lists for which that the node is a member. Presence in a large number of user-generated Twitter lists indicates the user’s popularity among Twitter users. The figure shows a high correlation of our computed node hierarchy with this measure of Twitter user popularity.

Finally, we measure the correlation with a popular twitter measure, Follower/Friend ratio, in Figure 2.4c. Popular users in Twitter tend to have an order of magnitude more followers than friends. We once again see a strong correlation between this measure and our computed hierarchy level.

### 2.5.2 Importance of Edge Direction

We now perform an experiment to validate that edge directions encode hierarchy information. For this, we use the college football dataset.

**College Football Dataset:** This dataset is constructed by looking at all (American) Football games played by College teams in Division 1 FBS (the highest division, formerly called 1-A) during five year period 2005-2009. The number of teams...
Figure 2.6: Hierarchy in the College Football network.

varies each year, but is between 150 and 200 for all five years. For each year, we consider
the win-loss record of these teams. In the graph, each team is a node, and we place an
edge from $u \rightarrow v$ if $v$ played and defeated $u$ during the season. We only consider the
win-loss records and do not consider the margin of victory.\textsuperscript{7} We also do not consider
other factors like home advantage, though these would lead to better predictions. We
end up with a directed unweighted graph representing win-loss record for a full season.

For each season, we find the optimal hierarchy. There is a lot of variation between
the quality of college football teams and we expect to see high hierarchy as observed in
Figure 2.6.

\textit{Random redirection:} Since the complete schedule is fixed before any games are
played, we can compare the hierarchy we observe in the directed graph to the hierarchy
if all games were decided by a random coin toss. In terms of the graph, this amounts to
redirecting each edge in the network randomly. This technique allows us to observe the

\textsuperscript{7}The margin of victory is not considered even in the official BCS computer rankings, since “running
up the scoreboard” is considered bad form and is discouraged.
effect of the directions on hierarchy once the undirected graph is fixed. This random redirection would eliminate any quality difference between the nodes, and we now expect to see a much smaller hierarchy in the redirected graph. To observe the variance of the random redirection, we repeat this experiment five times. The hierarchy for these randomly redirected graphs is also shown in Figure 2.6.

We see that the five randomly redirected graphs have very similar hierarchy, which is significantly lower than the real graph, showing that directions encode important information about hierarchy.

2.5.3 Hierarchy in Online Social Networks vs Random Graphs

To better understand how hierarchy emerges in a directed graph, we look at the behavior of hierarchy in random graphs to establish a baseline. We generate a random directed graph using the standard Erdös and Rényi [1960] random graph model as follows. We fix a probability \( p \) that will decide the density of the graph. For each ordered pair of vertices \((u, v)\), we put an edge from \( u \) to \( v \) with probability \( p \). The outdegree distribution of nodes in this graph is a binomial distribution where each node has expected degree \( np \).

![Figure 2.7: Hierarchy in random graphs.](image)

Figure 2.7a shows that, for random graphs, the hierarchy starts out being large, and monotonically decreases as the size of the graph increases. We can also see that
for small graph sizes, the variance is high, but as the graph size increases, the variance become very small.

We also conduct this experiment for different values of density, $p$. Figure 2.7b shows the outcome of the experiment with three different values of $p$. We see that for the same graph size $n$, hierarchy decreases with density. Hence, for random graphs, sparse graphs have higher hierarchy.

A plausible explanation for this phenomenon is that for sparse graphs, the randomness does not “cancel out”, but as the size of the graph and the density increases, the randomness “cancels out” and we get a low measure of hierarchy. It is not clear, that for a given the size of the graph $n$ and the density $p$, what the hierarchy will be and whether it will go to 0 as either $n$ or $p$ tends to infinity. We leave these questions as open directions for future research.

**Curated Lists on Twitter:** We now measure hierarchy for different online social networks. For this experiment, we collect curated lists on Twitter that correspond to different types of users.

**Famous people by field:** Similar to the journalists dataset described earlier, we collect curated lists of famous people in the fields of Technology, Journalism, Politics, Anthropology, Finance and Sports. The smallest collection is Anthropology with fifty nine people and the largest is Technology with almost three thousand people.

**Organizations:** We also look at lists of employees of different organizations that have a team presence on Twitter. These include forrst, tweetdeck, ReadWriteWeb, wikia, techcrunch, Mashable, nytimes and Twitter. The smallest graph, forrst, has just seven employees. The largest is Twitter with two hundred and eighty two employees.

For each of these lists, we reconstruct the Twitter graph restricted to just these nodes, i.e. the nodes in the restricted graph are all the people on a particular list and there is an edge between two nodes if there is an edge between them on Twitter. For
all these graphs, we calculate the hierarchy. Figure 2.8 shows a plot of hierarchy with respect to network size. We see that, among the fields, Sports has the highest hierarchy while Finance has the lowest one, and among organizations, the TODAYshow has the highest hierarchy while TweetDeck and ReadWriteWeb have the lowest one. Another trend that is observed is that, as the network size becomes larger, the hierarchy also increases. This is in contrast to random graphs, where the hierarchy decreases as the network size increases.

Wikipedia administrator voting dataset: Leskovec, Huttenlocher, and Kleinberg [2010b,a] collected and analyzed votes for electing administrators in Wikipedia. We use the wiki-vote dataset they collected and observe a very strong hierarchy in this dataset. This is consistent with the finding in [Leskovec, Huttenlocher, and Kleinberg, 2010a] that status governs these votes more than balance.

2.5.4 Effect of Scaling on Social Hierarchy

So far, we looked at small and medium sized graphs to get insight on how the measure of hierarchy works. We noticed that the hierarchy increases as the network size increases. Now, we shall consider large graphs to see the effect of scale on hierarchy in social networks.

For this experiment, we sample four popular directed social networks: Delicious, YouTube, LiveJournal and Flickr. The nodes are users and the edges indicate a follower relationship. We start from a single node and crawl nodes in the graph in a breadth first traversal. We plot hierarchy for different sizes of the graph. This is shown in Figure 2.9a.

We observe that, as an online social network grows in size, the hierarchy either stays the same or increases. This is in contrast with random graphs, where the hierarchy decreases as the graph grows in size. This suggests that, within small groups, social rank does not play an important role while forming connections but, as the group size
increases, social rank becomes important to people while forming links.

This result corresponds with the intuition that, in social networks, people form connections with others based on their perceived level in the social hierarchy.

Further, we see that different social networks have different amount of hierarchy: YouTube has the lowest hierarchy, Flickr and LiveJournal have medium hierarchy, and Delicious has the highest hierarchy.

*Number of strata:* Figure 2.9b plots the number of social strata in these four social networks, as we increase the graph size. We see that the number of strata stabilizes around seven for LiveJournal and around five for Flickr. YouTube has the lowest number of levels, and it also has the lowest hierarchy, while Delicious has the largest number of levels and also has the highest hierarchy. Compared to the number of nodes (100,000), the number of strata (<15) is very low.

*Rank distribution:* Figure 2.10a plots the frequency distribution of people belonging to different social strata in a network, i.e., how many nodes belong to each stratum. We see that, in all the networks, most nodes have a low rank in the hierarchy (between
one and three). A very small fraction of the nodes have ranks above four.

![Graph showing the effect of network size on hierarchy](image)

\(\text{(a) Effect of network size on value and variance of hierarchy}\)

\(\text{(b) Effect of network size on number of strata in the hierarchy}\)

**Figure 2.9: Effect of network size on hierarchy.**

The exception to this is Delicious, which has a wider distribution of ranks. We show the exact probability distribution of the Delicious nodes in Figure 2.10b. The plot shows that a lot of delicious nodes have medium ranks in the hierarchy. But, even in Delicious, very few nodes belong to the highest stratum.

![Graph showing the cumulative distribution of ranks](image)

\(\text{(a) Cumulative distribution of ranks}\)

![Graph showing the probability distribution for the Delicious graph](image)

\(\text{(b) Probability distribution for the Delicious graph}\)

**Figure 2.10: Distribution of ranks among nodes.**

*Agony distribution:* Our measure of hierarchy is based on the intuition that people prefer to connect to other people who are in the same stratum or higher up. People who connect to others lower in the hierarchy incur agony. Figure 2.11a plots the distribution...
of agony among the nodes in the different networks that we study. The figure shows that most people incur very small amount of agony. There are a few people who incur a lot of social agony. These people tend to follow a lot of people who are lower than them in the hierarchy.

**Random redirection:** We now study whether the hierarchy for each of these social networks is more or less than that observed in a randomly directed graph with the same underlying structure. To do this, we take each graph and randomly change the direction of each edge. Hence, we keep the undirected graph the same, but change the direction of the edge. In Figure 2.12, we show the importance of edge directions to hierarchy for these social networks and the effect of randomly redirecting the edges.

Among the social networks we studied, Delicious has the highest hierarchy. The networks starts out with medium hierarchy and it keeps increasing. The Delicious graph has almost perfect hierarchy at size 100,000. The hierarchy in the randomly redirected graph, shows a similar overall pattern but with low hierarchy. Delicious also
has the most number of levels in the hierarchy. YouTube, on the other hand, has the lowest hierarchy, which is even lower than the hierarchy observed if the edges were randomly oriented. The likely reason for this is that YouTube has a good search index and the preferred way of getting to videos is through search. Hence, social connections become less important and people do not connect to each other based on rank. In Flickr, the hierarchy largely remains the same even as the graph becomes large. However, the hierarchy in the redirected graph decreases sharply. In LiveJournal, the hierarchy starts out being very low and increases slowly with graph size. The randomly redirected graph on the other hand shows exactly the opposite behavior, consistent with the behavior of random graphs that we saw earlier.
2.6 Related Work

Early efforts to find the hierarchy underlying social interactions followed from observations of dominance relationships among animals. Landau [1951] and Kendall [1962] devised statistical tests of hierarchy for a society, but with the necessary assumption that there exists a strict dominance relation between all pairs of individuals, and that the relations are transitive (i.e. no cycles). Although de Vries [1995, 1998] expanded the Landau and Kendall measures by allowing ties or missing relationships, his algorithms are feasible only on small graphs.

The hierarchy underlying a social network can be used in recommending friends (the link prediction problem [Liben-Nowell and Kleinberg, 2003]) and in providing better query results [Kleinberg, 1999]. There exist link-based methods of ranking web pages [Getoor and Diehl, 2005]. Maiya and Berger-Wolf [2009] begin from the assumption that social interactions are guided by the underlying hierarchy, and they present a maximum likelihood approach to find the best interaction model out of a range of models defined by the authors. In the same vein, Clauset, Moore, and Newman [2006] use Markov Chain Monte Carlo sampling to estimate the hierarchical structure in a network. Rowe, Creamer, Hershkop, and Stolfo [2007] defined a weighted centrality measure for email networks based on factors such as response time and total number of messages, and tested their algorithm on the Enron email corpus. Leskovec, Huttenlocher, and Kleinberg [2010b,a] recently brought attention to signed network relationships (e.g. friend or foe in the Epinions online social network) and presented a way to predict whether a link in a signed social network is positive or negative.

The closest to our problem in the computer science literature is the minimum feedback arc set problem. In the minimum feedback arc set problem, we are given a directed graph $G$ and we want to find the smallest set of edges whose removal make the remaining graph acyclic. This is a well known NP-hard problem and is in fact NP-hard to
approximate beyond 1.36 [Kann, 1992]. Poly-logarithmic approximation algorithms are known for this problem [Even, Naor, Schieber, and Sudan, 1995].

2.7 Conclusions and Future Directions

In this chapter, we introduced a measure of hierarchy in directed social networks. We gave an efficient algorithm to find the optimal hierarchy given just the network. We also showed the emergence of hierarchy in multiple online social networks: in contrast to random networks, social networks have low hierarchy when they are small and the hierarchy increases as the network grows. We showed that there are a small number of strata, and this number does not grow significantly as the network grows.

We now give directions for future research. Our experiments in section 2.5.3 indicate that the measure of hierarchy is well behaved for random graphs. We leave open the question of finding closed form expression for hierarchy in random graphs. Another question we leave open is to find an algorithm that minimizes agony for sub-linear functions like \( f(u,v) = \log(r(u) - r(v) + 1) \). A different direction that we did not touch upon in this chapter is to study the emergence of hierarchy over time in a given social network. Finally, we could use this as a new signal and improve existing ranking algorithms.

This chapter dealt with the structure of social networks, and in particular exploiting the structure to obtain a measure of hierarchy. In the next chapter, we turn our attention to the dynamics of information flow over an underlying social network.
Chapter 3
News Propagation in Social Networks

3.1 Introduction

Online social networks have become an increasingly popular medium for sharing information such as links, news and multimedia among users. The average Facebook user has 120 friends, and more than 30 million users update their status at least once each day. More than 5 billion minutes are spent on Facebook each day (worldwide). As a direct consequence of these trends, social networks are fast overtaking traditional web as the preferred source of information.

In the early days of social networks, users tended to post predominantly personal information. Such information typically did not spread more than one hop, since only immediate friends were interested in it. Over time, online social networks have metamorphosed into a forum where people post information such as news that they deem to be of common interest. For example, during the recent Iran elections, traditional news media acknowledged the power and influence of social networks such as Twitter.

Prior work has studied various aspects of information sharing on social networks. Domingos and Richardson [2002, 2001] study the question of determining the set of

\[2\) http://siteanalytics.compete.com/facebook.com+google.com/?metric=uv
nodes in a network that will most efficiently spread a piece of information for marketing purposes. Kempe, Kleinberg, and Tardos [2003] proposed a discrete optimization formulation for this. Several recent studies focused on gathering intuition about influence spread from real-world data. Leskovec, Singh, and Kleinberg [2006] study the patterns of cascading recommendations in social networks by looking at how individuals recommend the products they buy in an online-retailer recommendation system. Leskovec, Backstrom, and Kleinberg [2009] develop a framework for the dynamics of news propagation across the web. Morris [2000] studied games where each player interacts with a small set of neighbors. He proved conditions under which the behavior adopted by a small set of users will spread to a large fraction of the network.

An aspect that has been overlooked so far is the question as to why users post information such as news or links on social networks. Unlike personal information, news typically propagates more than one hop, being received either from friends or from external sources, and reposted when found interesting. We argue that on receiving a news item, users behave strategically when weighting in various factors (such as how interested their friends will be in this news) to decide whether to further propagate that item by (re)posting it.

In this chapter, we posit that users in a social network have transitioned from being passive entities to strategic users. This trend leads to several interesting questions, such as: What factors do users consider when deciding whether to post an item? How does information diffuse over the social network based on different strategies used by the users? Furthermore, we envisage an advertising system where strategic users propagate ads to their friends on social networks. This is in contrast to the current model of ad dissemination on social networks where marketeers are given access to the social graph. We believe that a better understanding of user strategies in a social network is a key step towards such a system.

We use random graphs to formally model social networks. The formal definitions
of random graphs and the conditions under which they are connected are discussed in section 3.3. In section 3.5 we show that if we restrict attention to people who are interested in some particular news, then the induced subgraph retains the connectedness properties of the original graph. We prove our main result in proposition 3.10 which states that assuming strategic users, the spread of news over an online social network exhibits a threshold behavior. The news spreads to a significant fraction of the network if its quality is higher than a certain threshold that depends on how aggressive users are about posting news. If the quality is smaller than this threshold, only a sub-linear number of nodes in the network see the news.

The key contributions made in this chapter are:

1. We initiate the study of information propagation in social networks assuming strategic users.

2. We propose two models for strategic user behavior, greedy and courteous.

3. Assuming social networks can be modeled as certain random graphs, we prove that there is a threshold behavior when greedy users fully disseminate information.

4. We present a simulation study based on a real graph crawled from the Twitter social network, and show the threshold phenomenon holds in both strategic models of user behavior.

In what follows, we provide a detailed description of our results. We start by defining the user model.

### 3.2 Strategic User Model

We propose a simple game to model the behavior of users posting news on online social networks like Twitter and Facebook. For a particular user \( u \) in the network, whenever \( u \) first sees a previously unseen news item, she has the option of either posting it or not
posting it. Her utility is 0 if she does not post it. If she does, then her utility depends on (i) The set $I_u = \{\text{Neighbors who are interested in the news}\}$ and (ii) The set $S_u = \{\text{Neighbors who, u knows, have already seen the news before}\}$. Let $N_u$ denote the set of all u’s neighbors. We propose two particular forms for her utility:

1. **Greedy Strategy:** The utility is additive and for every neighbor who likes the news (irrespective of whether the neighbor has seen it before or not), she gets utility $+a$ and for every neighbor who does not, she gets utility $-b$. In this case, her decision to post only depends on $a, b$ and $f_u = \frac{|I_u|}{|N_u|}$. User u posts only if her utility is positive, that is, the fraction $f_u$ of users who like the news satisfies $\frac{af_u}{a+b} - \frac{b(1-f_u)}{a+b} > 0 \iff f_u > \frac{b}{a+b}$. Let us define $t = \frac{b}{a+b}$. In Section 3.4, we analyze this behavior and show that it depends critically on $t$.

2. **Courteous Strategy:** If a user does not post an item, then her utility is 0. The main difference from the greedy strategy is that the user does not want to spam her friends. We model this by saying that if more than a $c$ fraction of her friends have already seen the news before, she gets a large negative utility, when she posts the item. In case the fraction $\frac{|S_u|}{|N_u|} \leq c$, then her utility is the same as in the greedy case. In particular, she gets utility $+a$ for every neighbor who likes the news and has not seen it before (the set $I_u \setminus S_u$), and she gets utility $-b$ for every neighbor who does not like it (the set $I^c_u \setminus S_u$). Hence, her strategy in this case is to post if the fraction of neighbors who have seen the news $\frac{S_u}{N_u} \leq c$ and if the fraction $f_u$ of neighbors in $S^c_u$ who are interested in the news is $\geq t$. Note that, in this utility function, if a larger number of the user’s neighbors have posted the news, she is less likely to post it. In section 3.5, we show simulation results for this behavior on a small sample of the Twitter Graph.

---

$u$ might not know the true set of neighbors who have seen the news. She knows that a friend has seen the news only if a mutual friend posted it. This also means that we assume that every user knows which of her friends are themselves friends.
3.3 Random Graph Models of Social Networks

Erdős and Rényi [1960] started the study of random graphs. They consider a single parameter family of random graphs. The parameter $p$ corresponds to the density of the edges in the graph.

**Definition 3.1** (Erdős–Rényi random graph model). Given $n$ the number of vertices and a parameter $0 \leq p \leq 1$, a graph in this model is generated by adding an edge $(u, v)$ between vertices $u$ and $v$, independently at random with probability $p$. This process generates a probability distribution over all graphs which we denote by $G(n, p)$.

Initially, Erdős and Rényi random graphs were used as a model of real-world networks. With the advent of computational tools it became feasible to experimentally study large graphs and various properties about real-world networks were discovered. Travers and Milgram [1969] and Watts and Strogatz [1998] show the small-world effect in real-world networks which states that the diameter of the graphs is small. Barabási [2005] shows that the degree distribution of real-world networks follows a power law and hence has a heavy tail unlike for Erdős and Rényi random graphs. These properties made it clear that the Erdős and Rényi random graph model was not suitable as a model of real-world social networks.

Various models like small-world networks [Watts and Strogatz, 1998] and preferential attachment [Barabási and Albert, 1999] were proposed as models of social networks to overcoming the shortcomings of the Erdős and Rényi model. We focus on the stochastic Kronecker graph model proposed by Leskovec and Faloutsos. Leskovec and Faloutsos [2007] model social networks using stochastic Kronecker graphs and show that these graphs have various properties like a power law degree distribution, small diameter, densification power law ($E \propto N^k$) [Leskovec, Kleinberg, and Faloutsos, 2005] and shrinking diameter over time. We shall see the details of this model in 3.3.1.

$G(n, P)$ is a generalization of the well-studied model $G(n, p)$ [Erdős and Rényi, 1960]
as well as of stochastic Kronecker graphs [Leskovec and Faloutsos, 2007, Mahdian and Xu, 2007]. We can recover the \( G(n, p) \) model as well as the Kronecker graph model by putting conditions on the matrix \( P \). We shall prove our main results in section 3.4 in the general model \( G(n, P) \) of random graphs. We start with the formal definition of \( G(n, P) \).

**Definition 3.2** (Random Graph \( G(n, P) \)). Given a real symmetric matrix \( P \) with \( 0 \leq p_{i,j} \leq 1 \), denote by \( G(n, P) \) a random graph where edge \((i, j)\) exists with probability \( p_{i,j} \).

**Definition 3.3** (Weighted Projection of a Random Graph). Given a real symmetric matrix \( P \) with \( 0 \leq p_{i,j} \leq 1 \), we shall denote by \( W(P) = (V, E(P)) \) the weighted graph with \( P \) as its adjacency matrix. Note that \( p_{i,j} \) gives the weight of edge \((i, j)\) in \( W(P) \).

Lemma 3.4 gives conditions on the matrix \( P \) that ensure that the graph \( G \) sampled from the distribution \( G(n, P) \) is connected.

**Lemma 3.4** ([Mahdian and Xu, 2007]). For a large enough constant \( c \), if the weight of the min-cut in \( W(P) \) is at least \( c \log n \), then, with high probability\(^6\), the sampled graph \( G \sim G(n, P) \) is connected.

**Definition 3.5** (Induced Subgraph). Given a graph \( G = (V, E) \) and \( U \subseteq V \), we shall denote the subgraph of \( G \) induced by \( U \) as \( G[U] \) and the subgraph of \( W(P) \) induced by \( U \) as \( W(P)[U] \).

Before we begin, let us review the statement of the popular Chernoff Bounds. We use this a lot in the next sections. Proof of the bounds can be found in [Chung and Lu, 2006, Theorem 3.2].

**Theorem 3.6** (Chernoff Bounds). Let \( X_1, X_2, \ldots, X_n \) be independent random variables with

\[
Pr [X_i = 1] = p_i, \quad Pr [X_i = 0] = 1 - p_i
\]

\(^6\)Throughout this chapter, with high probability means with probability \( 1 - o(1) \).
Consider the random variable $X$ that is defined as the sum of these random variables.

$$X = \sum_{i=1}^{n} X_i.$$  The expectation of $X$ is given by $\mathbb{E}[X] = \sum_{i} p_i$. Then,

$$\Pr[X \leq \mathbb{E}[X] - \lambda] \leq e^{-\frac{\lambda^2}{2\mathbb{E}[X]}}$$

$$\Pr[X \geq \mathbb{E}[X] + \lambda] \leq e^{-\frac{\lambda^2}{2\mathbb{E}[X] + \lambda/3}}$$

### 3.3.1 Connectivity of stochastic Kronecker graphs

We shall prove our main results in section 3.4 in the general model $G(n, P)$ of random graphs. Stochastic Kronecker graphs are a good model of social networks. Since stochastic Kronecker graphs are a subset of the $G(n, P)$ model of random graphs, the results that we prove in section 3.4 apply directly to stochastic Kronecker graphs as well.

While the parameters that we obtain in the general case are not tight, we shall show that when we apply them to the stochastic Kronecker graphs, they become tight. In this section, we prove results that allow us to transfer results from the general model to stochastic Kronecker graphs. We begin with a definition of Kronecker multiplication of two matrices.

**Definition 3.7 (Kronecker product).** Given two matrices $A, B$, the Kronecker product or tensor product of $A$ and $B$ is defined as

$$A_{n \times m} \otimes B_{k \times l} = C_{nk \times ml} = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \ldots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \ldots & a_{2,m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}B & a_{n,2}B & \ldots & a_{n,m}B \end{bmatrix}$$

**Definition 3.8 (stochastic Kronecker graphs).** Given an initiator matrix $A$, where each element of $a_{i,j} \in [0, 1]$. Consider the matrix $P = A^\otimes k = A \otimes A \cdots \otimes A$. Then the random graph $G(n, P)$ is called a stochastic Kronecker graph. We shall denote such a graph by $K(A, k)$. 

[Leskovec and Faloutsos, 2007] model social networks using Kronecker graphs. They show that the graphs exhibit various real-world properties like a heavy-tail degree distribution, small diameter, densification power law \(E \propto N^k\) [Leskovec, Kleinberg, and Faloutsos, 2005] as well as shrinking diameter over time. They show that Autonomous Systems on the Internet, the citation graph for high energy physics from arXiv and the U.S. Patent citation database can be modeled well using \(G(n, P)\) where \(P = A^{\otimes k}\) for particular \(2 \times 2\) initiator matrices \(A\) of the form \(\begin{bmatrix} \gamma & \beta \\ \beta & \alpha \end{bmatrix}\) and \(\alpha \geq \beta \geq \gamma\) and a given integer \(k\).

[Mahdian and Xu, 2007] give tight conditions on the connectivity of \(K(A, k)\) in terms of properties of the initiator matrix \(A\). Lemma 3.9 states these conditions.

**Lemma 3.9.** For fixed constants \(\alpha \geq \beta \geq \gamma\), the necessary and sufficient conditions for the Kronecker graph \(K(A, k)\) to be connected with high probability is \(\beta + \gamma > 1\) or \(\alpha = \beta = 1, \gamma = 0\).

In fact, given a constant \(c\), as \(n\) becomes large enough, if \(\beta + \gamma > 1\), the size of the min-cut is \(> ck(= c \log n)\).

In the next section, we shall prove results on the spread of news when the underlying random graph \(G\) is sampled from \(G(n, P)\) and \(P\) satisfies some special characteristics. Lemma 3.9 allows us to transfer results that we prove for graphs \(G(n, P)\) where min-cut of \(W(P) > c \log n\) to connected Kronecker graphs.

### 3.4 Analysis of a Model of Strategic User Behavior

In this section we analyze the spread of news in a social network. We start with a high level description of the proof that news propagation in social networks follows a threshold behavior. We prove our results in the \(G(n, P)\) model of social networks defined in section 3.3. Lemma 3.4 gives a condition on the matrix \(P\), in terms of the min-cut of \(W(P)\), under which a random graph \(G\), sampled from \(G(n, P)\), is connected.

\(^7\)In this chapter, \(A\) will always be a \(2 \times 2\) matrix of the form \(\begin{bmatrix} \gamma & \beta \\ \beta & \alpha \end{bmatrix}\) where \(\alpha, \beta, \gamma\) are fixed constants and \(\alpha \geq \beta \geq \gamma\).
We assume that users follow the *greedy* strategy defined in section 3.2. A user playing the greedy strategy posts only if the fraction of interested neighbors is larger than the threshold $t$. The greedy strategy is monotone in the set of neighbors who are interested and have posted the news. This means that the probability of posting increases with the set of friends who have posted the news. Hence, to analyze the set of users who post the news under the greedy strategy we do not need to analyze the dynamic sequence of users that post the news but we only need to analyze the static subgraph of users interested in the news and who initially receive the news.

We assume that for a given news, each user in the network likes it with probability $q$, which is independent of everything else.\(^8\) The probability $q$ is just a number $\in [0, 1]$ that could model the quality of the news item or the inherent interest the subject generates. If we restrict attention to the subgraph of users that are interested in the news, then the induced subgraph is a random sample of the vertices of the graph $G$. Lemma 3.12 shows that the min-cut condition is preserved for a subgraph induced by a random sample of the vertices. Hence, the same connectedness properties hold for the induced subgraph. Finally, in proposition 3.10 we show that if the quality $q$ of the news is larger than the threshold $t$, then all users interested in the news also post it, if the news reaches them. Hence, the induced subgraph is connected with high probability and the news spreads across the network.

We start with some definitions. Color nodes in $G$ not interested in the news blue and the ones interested in the news yellow. Call the set of blue vertices $B$ and yellow vertices $Y$. A yellow node is *responsive* if more than a $t$ fraction of its neighbors are interested in the news. *Responsive nodes* are precisely the ones in the graph that will repost the news if they receive it. Color responsive nodes red and call this set of nodes $R$. Hence, we have the following set of nodes:

\(^8\)It is an interesting question to relax this assumption, since in a typical social network we might expect nodes with similar interests to be clustered together.
$B$ : The subset of nodes of $G$ that are not interested in the news.

$Y$ : The subset of nodes of $G$ that are interested in the news.

$R$ : The subset of nodes of $G$ that are interested in the news and will post is if they receive the news. Note that $R \subseteq Y$.

Recall that for a subset $S \subseteq V$, $G[S]$ denotes the subgraph of $G$ induced by the vertex set $S$. We are interested in the structure of the graph $G[R]$. Throughout, we assume that $q$ and $t$ are constants that do not depend on the number of nodes $n$. We prove the following results in the next two sub-sections:

**Proposition 3.10.** Suppose $P$ is a matrix such that the min-cut of the weighted graph $W(P)$ had weight $\geq c \log n$, for a large enough constant $c$. Suppose $G$ is a random graph sampled from the distribution $G(n, P)$ and that $\log n$ random nodes in the $G$ initially see the news.

If $q > t$, then, with high probability, almost all nodes interested in the news will see it. On the other hand, if $q < t$ then only a sub-linear number of the vertices will post the news.

**3.4.1 The Connectedness of $G[R]$**

We prove Proposition 3.10 in this subsection. We first show that $G[R]$ is connected and that its size is a constant fraction of the size of $G$. Then, we show that, if the news is randomly received by a non-negligible number ($\log n$) of users in $G$, someone in $G[R]$ will get it and hence it will spread to all nodes in $G[R]$. Let $\rho$ be a large enough constant that satisfies the condition of lemma 3.4.

Throughout this section, we shall assume that $P$ is such that the min-cut of $W(P)$ has weight $\geq c \log n$, where $c = \max \left\{ \frac{16\rho}{q^2}, \frac{8q}{(t - q)^2} \right\}$.

We have two sources of randomness here: For every edge $(i, j)$ in $G$ we have a coin which decides whether edge $(i, j) \in G$. We call this set of coins $C_E$. Also, for every
vertex \( v \in V \), we have a coin toss that decides if \( v \in Y \). We call this set of coins \( C_V \). The probability space we have is over the union of the coins \( C_E \cup C_V \).

The next two lemmas show that the min-cut in the sampled graph \( G[Y] \) remains high.

**Lemma 3.11.** All nodes in the graph \( G \) have degree at least \( \frac{c}{2} \log n \), with high probability over coin tosses \( C_E \).

Define the random variable \( X_{uv} \) to take value 1 with probability \( p_{uv} \) and 0 with probability \( 1 - p_{uv} \). For a given vertex \( v \), the degree of \( v \) in \( G \) is given by \( X_v = \sum_u X_{uv} \). Using Chernoff Bounds, we can see that, for every constant \( \delta \),

\[
Pr \left[ X_v \leq (1 - \delta)\mathbb{E}[X_v] \right] \leq 2 \exp(-\mathbb{E}[X_v] \delta^2/2)
\]

\[
\leq 2 \exp(-c \log n \cdot \delta^2/2)
\]

\[
= n^{-c\delta^2/2}
\]

The above result is true for all \( \delta \). We choose \( \delta = \frac{1}{2} \).

\[
Pr \left[ X_v \leq \frac{1}{2} \mathbb{E}[X_v] \right] \leq n^{-c/8}
\]

\[
\leq n^{-2} \ (\text{Since } c \geq 16)
\]

Apply union bound to get that for all vertices in \( G \), with probability \( > 1 - 1/n \), the degree of each vertex is \( > (1 - \delta)c \log n = \frac{c}{2} \log n \) \( \square \)

The next lemma shows that if we sample nodes independently from the weighted graph \( W(P) \), then the weight of the min-cut in the induced subgraph \( W(P)[Y] \) is large.

**Lemma 3.12.** The weight of the min-cut in \( W(P)[Y] \) is at least \( \rho \log n \), with high probability over the coin tosses \( C_V \).

To prove this lemma, we use Chernoff bounds to upper bound the probability that the cut \( (S, V \setminus S) \) in the induced subgraph is small. Then we use union bound
over all cuts. However, we cannot apply the union bound directly. We partition all cuts depending on the weight of the cut. The Chernoff bound in weaker for cuts of small weight, but the number of cuts of small weight is also small. Hence, we partition all cuts depending on their weight, and then apply Chernoff bound to limit the probability that there is a small min-cut in \( W(P)[Y] \) due a cut of a particular weight in \( W(P) \).

Call a cut \( \alpha \)-minimal if the size of that cut is within an \( \alpha \) factor of the size of the min-cut. We use the theorem [Karger and Stein, 1996, Theorem 8.4] that bounds the number of weighted \( \alpha \)-minimal cuts: In any weighted graph, the number of \( \alpha \)-minimal cuts is \( O\left((2n)^{2\alpha}\right) \). Let the weight of the min-cut in \( W(P) \) be \( m \). We partition all cuts depending on their size. For a integer \( k \), all cuts with weight in \( [mk, m(k + 1)] \) belong to the same partition. All these cuts are \((k + 1)\)-minimal, hence, there are at most \( O\left((2n)^2k\log n/2\right) \) of them.

Consider any cut \((S, V \setminus S)\) having weight \( t \in [mk, m(k + 1)] \). Let \( X \) be the random variable which gives the weight of the cut in \( W(P)[Y] \). Each edge of \( W(P) \) is also an edge of \( W(P)[Y] \) with probability at least \( q^2 \), hence \( \mathbb{E}[X] \geq tq^2 \). By Chernoff Bounds, for all \( \delta \geq 0 \), \( \Pr[X \leq (1 - \delta)q^2] \leq \exp\left(-\delta^2tq^2/2\right) \leq \exp\left(-\delta^2q^2km/2\right) \leq \exp\left(-\delta^2q^2kc\log n/2\right) = n^{-\delta^2q^2kc/2} \).

Now, applying the union bound over all cuts, we get that with probability \( p = 1 - \sum_k n^{(-\delta^2q^2c/2)k(2nq^2)^{2k+2}} \) all cuts in the subgraph \( W(P)[Y] \) have min-cut of size \( \geq (1 - \delta)q^2 \) Size of min-cut in \( W(P) = (1 - \delta)q^2c\log n \) with high probability.

Choose \( \delta = \frac{1}{2} \). By assumption, \( c \geq \frac{16\rho}{q^2} = \frac{4\rho}{\delta^2q^2} \).

Substituting these values, we get that the size of the min-cut in \( W(P)[U] \) is \( > \rho \log n \) with high probability.

Suppose we toss all coins in \( C_V \cup C_E \), but do not reveal them all. First we reveal only the coins in \( C_V \). Lemma 3.12 shows that \( W(P)[Y] \) has a min-cut of size \( \rho \log n \) with high probability. Now, we reveal coins in \( C_E \). Lemma 3.4 shows that, conditioned on \( C_E \) and
that $W(P)[Y]$ has a min-cut of size $\rho \log n$ the graph $G^Y$ sampled from $G(|Y|, W(P)[Y])$ is connected. Now, applying union bound, we get that $G[Y]$ is connected, with high probability over $C_E \cup C_V$.

We shall now prove the main theorem of this section. We prove that $G[R]$ is connected by using the fact that the min-cut of $G$ is large.

**Theorem 3.13.** If $q > t$ then, with high probability, every vertex in $Y$ also belongs to $R$ and so $G[R]$ is connected. On the other hand, if $q < t$, then $G[R]$ only contains $o(n)$ vertices, with high probability.

**Proof.** To prove this, we first reveal the coin tosses $C_E$. That gives us a graph $G$ sampled from the distribution $G(n, P)$. Lemma 3.11 shows all nodes in $G$ have degree $\geq \frac{c}{2} \log n$ with high probability over $C_E$. We assume for the rest of the proof that $G$ is a fixed graph with these properties.

Now, we reveal the coins $C_Y$. This corresponds to a random process of coloring nodes of $G$ yellow with probability $q$. Let $Y_q$ be a random variable that takes value 1 with probability $q$ and 0 with probability $1 - q$. For a node $v$, let $d(v)$ denote its degree in $G$. We analyze two cases.

$q > t$:

$$\Pr [v \notin R] = \Pr \left[ \sum_{i=1}^{d(v)} Y_q < t \cdot d(v) \right]$$

$$= \Pr \left[ \sum_{i=1}^{d(v)} Y_q < qd(v) - (q - t)d(v) \right]$$

$$\leq \exp \left( - \frac{((q - t)d(v))^2}{2qd(v)} \right)$$

$$\leq n \cdot \frac{(t - q)^2 c}{4q}$$

The third inequality follows from Chernoff Bounds, since $Y_q$ are independent and $\mathbb{E}[Y_q] = q$. The last is true since $d(v) \geq \frac{c}{2}\log n$. We apply the union bound to get
that with probability $\geq 1 - n \cdot n^{-\frac{(t-q)^2}{4q}} \geq 1 - \frac{1}{n}$ (since $c > \frac{8q}{(t-q)^2}$), all nodes in $Y$ also belong to $R$.

Now, we apply union bound to get that $Y = R$, with high probability over the event $C_E \cup C_V$.

Since, we proved that $G[Y]$ is connected with high probability over $C_E \cup C_V$, we get that $G[R]$ is connected with high probability over the event $C_E \cup C_V$.

$q < t:$

$$Pr [v \in R] = Pr \left[ \sum_{i=1}^{d(v)} Y_q \geq t \cdot d(v) \right]$$

$$= Pr \left[ \sum_{i=1}^{d(v)} Y_q \geq qd(v) + (t-q)d(v) \right]$$

$$\leq \exp -\frac{((q-t)d(v))^2}{2(qd(v) + (q-t)d(v))/3}$$

$$\leq n^{-\frac{(t-q)^2}{4q} \frac{3}{2q}}$$

Hence, $\mathbb{E}[|R|] \leq |Y| n^{-\frac{(t-q)^2}{4q} \frac{3}{2q}} \in o(n)$. So, $R$ only contains a sub-linear number of nodes of $G$.

\[\square\]

**Proposition 3.14.** Suppose $P$ is a matrix such that the min-cut of the weighted graph $W(P)$ had weight $\geq c \log n$, for a large enough constant $c$. Suppose $G$ is a random graph sampled from the distribution $G(n, P)$ and that $\log n$ random nodes in the $G$ initially see the news.

If $q > t$, then, with high probability, almost all nodes interested in the news will see it. On the other hand, if $q < t$ then only a sub-linear number of the vertices will post the news.

**Proof.** If $q < t$ then the proposition follows directly from theorem 3.13. When $q > t$, theorem 3.13 tells us that $G[R]$ is connected. If any node in $R$ receives the news it will
be propagated to all the nodes. But, the probability that none of the nodes in \( R \) get the news is \( \leq (1 - q)^{\log n} = O(n^{-q}) = o(1) \). Hence, with high probability, almost all the nodes interested in the news actually receive it.

In the next section, we shall look at a real-world graph and simulate the greedy and courteous behavior on this graph.

### 3.5 Simulation Results

In this section, we present results from the simulation of our two strategic user models over a partial crawl of the Twitter social network.\(^9\)

**Dataset:**

The dataset is obtained by means of a partial crawl of the Twitter social network. Since we are interested in link postings by typical users, we remove hubs (users with more than 500 friends) from our graph. The resulting graph we use in our experiments consists of 5978 users with 110152 directed friendship edges. Each simulation starts with a set of seed nodes (of size \( \log n \)), and in each round, nodes decide whether to post the link using one of the two models described earlier in the chapter.

**Results:**

We define coverage as the fraction of interested users who get to see the news item. Figure 3.1 plots how the coverage varies for the greedy and courteous strategies. For the courteous strategy we fixed \( q = 0.5 \). Each simulation was repeated 10 times, and the standard deviations are indicated in the form of error bars.

---

\(^9\)All the data was obtained in accordance with Twitter’s terms of use.
(a) Greedy strategy: The coverage exhibits a step behavior when $q = t$.

(b) Courteous strategy. ($q = 0.5$). The coverage decreases logarithmically as the user is more courteous, i.e. as parameter $c$ decreases.

Figure 3.1: Coverage of greedy and courteous strategies over the crawled graph. $q$ is the probability with which each user likes the link. $t$ is the threshold for the greedy strategy. $c$ is the threshold for the courteous strategy.
**Greedy Strategy:** Figure 3.1a shows that, for all values of \( q \), the coverage exhibits a step behavior and the step happens around \( q = t \). For different values of \( q \), the percentage of coverage decreases with \( q \). This is true even when \( t = 0 \), which means that the size of the connected components drops when we sample the graph. The densification power law \( (E \propto N^k) \) [Leskovec, Kleinberg, and Faloutsos, 2005] that has been observed in other social networks would predict this behavior.

**Courteous Strategy:** Figure 3.1b shows the effect of the courteous strategy on the coverage. The parameter \( c \) indicates the threshold of neighbors that have already seen the link. A courteous user posts the link only if less than \( c \) fraction of neighbors have already seen the link. The figure shows that the coverage decreases logarithmically as the user is more courteous. This result shows that even when the users are courteous if \( q > t \) then, news can still reach a reasonable fraction of the graph.

### 3.6 Conclusion and Future Work

We proposed the model of strategic user behavior in online social networks for news posting, defined two user models (greedy and courteous), presented formal analysis of the greedy model and simulated both models on the data set we collected from Twitter. We propose the following directions:

*Mine the Twitter data set:* Search for patterns in the way users post news links in order to validate the model and provide further insights about user strategies over social networks.

*Analyze the Courteous Strategy and Multiple Strategies:* We leave open a formal proof that the courteous strategy also exhibits threshold behavior. Further, we want to test whether similar results hold in a social network in the presence of multiple user strategies.
Design a framework for advertisement on online social networks of strategic users:

We believe that the strategic user model has applications in advertising and marketing. We plan to investigate incentive schemes for marketeers to encourage strategic users to advertise products over social networks.
Secretary Problems
Chapter 4

Secretary Problems on Independence Set Systems

4.1 Introduction

Consider the following problem: We want to hire one secretary for a job. Secretaries come in one by one and we can interview them. Assume that there is a strict linear ordering among all secretaries that is they are all comparable to each other and there are no ties. We want to select the best secretary. This is trivial in the offline setting. However, in the online setting where we are restricted to either make a job offer or reject the secretary when a secretary comes in, the problem become challenging. If we reject the secretary, the person never comes back.

It is clear that we cannot always select the best secretary in the online setting. In fact, if we adopt the worst case adversarial analysis typically used for online algorithms, we can always make up the input such that the any algorithm will always select the worst secretary! Surprisingly, Dynkin [1963] showed that, if we assume that the order in which the secretaries come is random, we can actually select the best secretary with constant probability $= \frac{1}{e}$.

Online Algorithms

Online algorithms are defined as algorithms where the entire input is not available to the algorithm at the start. In this chapter, we look at the problem where we are given a set of elements, and we need to select a subset that maximizes an objective function. In this setting, elements appears one after the other and the algorithm needs to make
irrevocable decisions about whether to accept the element or not.

Not knowing future input is a handicap, and so, in general online algorithms cannot choose a solution which is as good as that found by an offline algorithm. The performance of online algorithms is commonly measured in terms of the competitive ratio which measures how the performance of the online algorithm compares with the solution of the optimal offline algorithm. The competitive ratio of an algorithm $A$ is defined as the worst case ratio of the performance of the algorithm relative to the optimal offline algorithm $OPT$. If $v(A, i)$ is a value function which measures how good an input $i$ is for algorithm $A$, then the competitive ratio is given by:

$$\max_{i \in \text{inputs}} \frac{v(OPT, i)}{v(A, i)}$$

Note that according to this definition, the best competitive ratio an algorithm can achieve is 1, which occurs when the online algorithm always selects the best possible result. Typically, the competitive ratio is strictly greater than one. We say that an algorithm $A$ is constant competitive if there exists a constant $c$ such that $A$ has competitive ratio less than or equal to $c$.

### 4.1.1 The Random Permutation Model

There are different models of how input is presented to an online algorithm. The strictest one is adversarial, where the input is chosen adaptively by the adversary based on the choices made by the algorithm so far. Assuming a powerful adversary often leads to unreasonably pessimistic conclusions about online algorithms. Another shortcoming of looking only at the worst case behavior is that, even if two algorithms have the same worst case behavior, one might have much better performance for most inputs, which the other might exhibit the same behavior on all inputs and the analysis wont be able to distinguish between the two.
A much more benign assumption is that the input comes is sampled from a distribution. Distributional assumptions lead to better algorithms, however, these are generally too strong for most practical scenarios as most real world data does not satisfy these assumptions.

The random permutation model walks a fine line between these two. In this model, the input is still assumed to be the worst case input. However, the order in which it is presented to the algorithm is random. Hence, the worst case input is adversarially decided, then a random permutation on this is generated and the input is presented to the algorithm in the resulting order. We shall use this random permutation model in this chapter. This setting has come to be known as the secretary setting due to the problem discussed in a nice survey by Ferguson [1989] that we saw earlier. In this chapter, we generalize the secretary problem to Independence Set Systems. We start by looking at the definition of Independence Set Systems.

4.1.2 Independence Set Systems

An independence set system \( \mathcal{H} = (V, \mathcal{I}) \) is an ordered pair where \( V \) is the set of vertices and \( \mathcal{I} \) is a set of subsets of \( V \) such that

1. \( \phi \in \mathcal{I} \)

2. \( \forall S', S \text{ where } S' \subseteq S : S \in \mathcal{I} \implies S' \in \mathcal{I} \)

Members of \( \mathcal{I} \) are called independent subsets of \( \mathcal{H} \). Property 2 is called the hereditary property or down-monotonicity and so, these systems are also called hereditary set systems or down monotone set systems.

Independence Set Systems are a generalization of various structures like graphs, matroids, knapsacks, etc. Figure 4.1 shows an example of encoding the knapsack constraints as an independence set system.
4.1.3 The Secretary Problem on Independence Set Systems

We consider the problem of selecting a maximum independent set of vertices from an independence set system. In this section, we consider the setting where elements of $V$ do not have any weight. We shall extend it to the setting of elements with weight in Section 4.4. Formally, the problem is described as follows. We are given an independence set system $H = (V, I)$. A random permutation $\sigma : [n] \to V$ is generated and elements of $V$ are presented one by one in the order $\sigma(1), \sigma(2), \ldots, \sigma(n)$. At time step $i$, we are presented with a vertex $v = \sigma(i)$ and we need to make a decision whether to select $v$ or reject $v$. Once a vertex is selected, it cannot later be dropped. Conversely, if $v$ is rejected, then we cannot come back and select it. Hence, the decision to select or reject is irrevocable. The task is to select the maximum number of elements such that they are all independent in $H$.

Let $n$ be the number of elements $|V|$. Babaioff, Immorlica, and Kleinberg [2007b] gave an example of an independence set system showing that it is not possible to get
a $o\left(\frac{\log n}{\log \log n}\right)$ approximation algorithm to solve this problem even ignoring computational considerations. The set system is constructed by randomly throwing $n$ balls into $n$ bins. The independence set system is obtained by treating the bins as maximal independent sets.

**Theorem 4.1.** [Babaioff, Immorlica, and Kleinberg, 2007b] The expected size of the maximum set in $I$ is $\Omega \left(\frac{\log n}{\log \log n}\right)$. Suppose the elements of $V$ are presented in random order. For any online randomized algorithm $A$, the expected number of elements selected by $A$ is at most two.

Given the lower bound we explore the question of obtaining an approximation algorithm which matches this.

### 4.1.4 Related Work

The secretary problem was introduced into the computer science community though its applications to auctions [Kleinberg, 2005, Hajiaghayi, Kleinberg, and Parkes, 2004, Babaioff, Immorlica, Kempe, and Kleinberg, 2008].

Babaioff, Immorlica, and Kleinberg [2007b] introduce the Matroid Secretary Problem. In this setting, we are given a set of elements that form a matroid. Elements have weights, and we want to select an independent set of elements that has the maximum weight. This encompasses problems like finding the maximum weight forest in a weighted undirected graph. They give a $O(\log n)$ approximation to the general problem. They also give constant factor approximation algorithms for various special cases including graphical matroids (maximum weight forest). Dimitrov and Plaxton [2008] give a constant factor approximation for transversal matroids. Later, Korula and Pál [2009] gave a simpler algorithm and also improved the approximation ratio. Sato [2011], Gharan and Vondrák [2011] gave a constant-factor competitive algorithms for general matroids in the random assignment model where the weights are assigned randomly to
the elements of a matroid, and then the elements arrive on-line in an adversarial order.

Babaioff, Dinitz, Gupta, Immorlica, and Talwar [2009] consider the secretary problem with time discounting. Babaioff, Immorlica, Kempe, and Kleinberg [2007a] give a constant factor approximation for the case when elements have a weight and a value, and they must satisfy knapsack constraints. Kumar, Lattanzi, Vassilvitskii, and Vattani [2011] consider the problem of selecting a maximal element when there is only a partial order among the elements.

Variants where we want to maximize not just the number of sum of weights of the set of selected secretaries, but some general submodular function of the set of selected secretaries have also been studied [Feldman, Naor, and Schwartz, 2011, Bateni, Hajiaghayi, and Zadimoghaddam, 2010, Gupta, Roth, Schoenebeck, and Talwar, 2010]. Feldman, Naor, and Schwartz [2011] give constant factor approximation for maximizing general submodular functions for the case of uniform and partition matroids and under knapsack constraints. Bateni, Hajiaghayi, and Zadimoghaddam [2010] give a constant competitive algorithm for uniform matroids where the submodular function can be non-monotone. They also give a $O\left(l \log^2 r\right)$-competitive algorithm for selecting under $l$ matroid constrains where $r$ is the maximum rank of the matroids. They also give an $O\left(l\right)$-approximation algorithm when the feasible sets must satisfy $l$ knapsack constraints. Gupta, Roth, Schoenebeck, and Talwar [2010] give a polynomial time algorithm for the offline problem of maximizing non-monotone submodular functions on p-independence set systems. They specialize their algorithms for maximizing non-monotone submodular functions on matroids to obtain a constant factor approximation algorithm for uniform and partition matroids and a $O\left(\log k\right)$ algorithm for general matroids of rank $k$.

Since, Independence Set Systems are generalizations of structures like graphs, knapsack constraints, matroids, the secretary problem on these structures can be treated as special cases of the problem we solve. The approximation ratio of most of the special
cases is in general better. In this chapter, we only consider maximizing the number of elements and the weighted sum of selected elements. We do not consider more general functions, on the selected set of elements.

4.1.5 Bipartite Graph representation of Independence Set Systems

One way to represent an independence set system is as a bipartite graph $G = (L \cup R, E)$. The elements in the left set $L$ are called points and represent the vertices $V$ from the set system and the right elements from the right set $R$ are called blocks and these represent subsets of the vertices $V$ that are elements of $I$. We want to select a large subset of points in $L$ such that they share a common neighbor in $R$ i.e. we want to select a large independent set in $V$. Figure 4.2 shows a simple independence set system and its representation as a bipartite graph.

4.2 Oracles and Problem Variants

Our algorithm takes an independence set system as input. However, the number of independent sets in general set systems can be exponential in the number of vertices. Just specifying the set system can take exponential time. Hence, we assume oracle access to the set system. When a new vertex is presented the set system induced by all vertices seen so far is revealed, through some kind of an oracle. Starting from a very
simple oracle we now describe successively stronger oracles that we will use to solve the problem. The oracles can only process subsets of vertices that have arrived. (Also see Section 4.5).

$O_0$: This is the simplest oracle. Given a set of vertices $O_0$ tells if they are independent or not. In terms of the bipartite graph, given a set of left points, $O_0$ returns a right block that all the left points have in common if one exists otherwise return that no such block exists. With this oracle even the offline case of selecting a largest independent set is NP-hard. Moreover, it is even hard to approximate as seen by the special case of selecting an independent set of vertices from a graph.

$O_1$: To get over this and not restrict ourselves to a particular problem, we need access to an oracle which will give us the optimal solution to the offline problem. The simplest oracle is the one that solves the offline problem. It takes as input a set of elements $S$, and returns a maximum independent subset $T$ of $S$. If there are multiple maximum independent subsets, it arbitrarily chooses one. In terms of the bipartite graph, given a set $S$ of points in $L$, $O_1$ returns a block $r \in R$ with the maximum number of neighbors in $S$. If there are multiple elements in $R$ it arbitrarily chooses one. Note that access to $O_1$ only allows us to solve the offline problem. Using this to solve the online problem is not obvious. In fact, this oracle is still too weak to solve the problem. We propose a slightly stronger version of this next.

$O_2$: It takes as input two sets of elements $S, T$ such that $S \subset T$ and $S \in \mathcal{I}$. The oracle $O_2$ returns a maximum independent subset $W$ such that $S \subset W \subset T$. $O_2$ ensures that we select all the elements of $S$ and among all independent subsets of $T$ which also include $S$, we choose the largest one. If there are multiple maximum independent subsets, it arbitrarily chooses one. In terms of the bipartite graph, given a set of points $S, T$ such that there is a block $r \in R$ adjacent to all points in $S$, $O_2$ returns a block $p \in R$ with the maximum number of neighbors in $T$ among all blocks such that all elements in $S$ are adjacent to $p$. If there are multiple such blocks, $O_2$
arbitrarily chooses one.

\( \mathcal{O}_3 \): We need an even stronger oracle which we state now. We state this only in terms of the bipartite graph representation of an independence set system. Given this bipartite graph the oracle \( \mathcal{O}_3 \) takes as input a subset \( T \) of \( L \) and a threshold \( t \) and returns a right block \( r \in R \) uniformly at random from all right blocks in \( R \) that connect to all points in \( T \) and have degree \( \geq t \).

It is easy to show that for \( i \in \{0, 1, 2\} \), oracle \( \mathcal{O}_i \) can be simulated by at most \( O(\log n) \), calls to oracle \( \mathcal{O}_{i+1} \). Hence, we assume that if we are given \( \mathcal{O}_1 \), then we are also given \( \mathcal{O}_0 \). If we are given \( \mathcal{O}_2 \), then we are also given both \( \mathcal{O}_1 \) and \( \mathcal{O}_0 \). If we are given \( \mathcal{O}_3 \), then we are also given \( \mathcal{O}_2, \mathcal{O}_1 \) and \( \mathcal{O}_0 \). We solve the secretary problem on independence set systems using the oracle \( \mathcal{O}_3 \). We give an algorithm which outputs a set of size \( \Omega \left( \frac{s}{\log n \log \log n} \right) \), where \( s \) is the size of the largest independent set (largest right degree in \( R \)). This gives a \( O(\log n \log \log n) \) approximation algorithm, which matches the lower bound within \( O((\log \log n)^2) \) factors. We leave the problem of finding a large independent set using the simpler oracle \( \mathcal{O}_2 \) open. A few directions exploring this oracle further are presented in Section 4.5.

### 4.3 Algorithm

In this section we shall describe the actual algorithm for the unweighted case. The algorithm starts by selecting a random sample of the points (elements of \( L \)) of size half the total number of points. It’s possible to do this since the input comes in a random order. This happens in step 2 of the algorithm. The random sample of points is used to select a threshold size such that we reject all sets smaller than that size. This is done in step 3. Finally, \( \mathcal{O}_3 \) is used to select an independent set from the sample (a block from \( R \)). This is selected such that the number of neighbors of the block in the sample is large enough, but also that it has enough “space” left for new points. For each new point, it is selected if and only if it belongs to the selected block. This is done in step 4.
We now formally describe the algorithm.

**Input**: A graph $G = (L \cup R, E)$ where access to $E$ is given by the edge oracle $E$ which takes as input a subset $T$ of $L$ and returns true if all vertices in $T$ share a common neighbor. We also assume we have access to the oracle $O_3$ (along with $O_2, O_1$ and $O_0$) mentioned before. Points in $L$ are seen one by one in random order.

**Output**: A set $S \subset L$ of selected points that all have a common right block as a common neighbor.

**Algorithm**:

1. Set $S \leftarrow \phi$.

2. Sample an integer from the binomial distribution. $k \sim \text{Binom}(n, 1/2)$. Look at the first $k$ points and don’t select any. Call this set of points $F$ (first half). This is the random sample.

3. Among the points seen so far ($F$), use the oracle $O$, to find the largest degree of a right block using $O_2$. Call this number $\hat{s}$. Choose $i \in [0, 8 \log \log n]$ and $\alpha \in \left\{ \frac{5}{2}, \frac{5}{2} \left( \frac{5}{7} \right), \frac{5}{2} \left( \frac{5}{7} \right)^2, \frac{5}{2} \left( \frac{5}{7} \right)^3 \right\}$ uniformly at random. Set threshold $t = \frac{\alpha \hat{s}}{2^i} = \frac{\alpha \hat{s} \cdot s}{s \cdot 2^i}$.

4. Use the oracle $O_3$ to get a right block $r = O_3(F, t)$. For every new point $v$, if $v$ is connected to $r$, $S \leftarrow S \cup v$

We begin with a lemma that explains the choice of the set that $\alpha$ belongs to in step 3.

**Lemma 4.2.** Let two intervals $[a, b]$ and $[c, d]$ such that $a < b$ and $c < d$. Then the set $S = \left\{ \frac{c}{a} \left( \frac{c}{d} \right)^i \right\}_{0 \leq i \leq m}$ is such that for any $k \in [a, b]$, there exists an element $s \in S$ such that $k \cdot s \in [c, d]$. The size of $S$ is at most $\left\lfloor \frac{\log \frac{b}{a}}{\log \frac{c}{d}} \right\rfloor$. 

Proof. Consider the partition of \([a, b]\) into sub-intervals \([a, \frac{ad}{c}] \cup [\frac{ad}{c}, \frac{ad^2}{c^2}] \cup [\frac{ad^2}{c^2}, \frac{ad^3}{c^3}] \cup [\frac{ad^3}{c^3}, \frac{ad^4}{c^4}] \cup [\frac{ad^4}{c^4}, \frac{ad^5}{c^5}] \cup \ldots \cup [\frac{ad^{m-1}}{c^{m-1}}, \frac{ad^m}{c^m}]\). For a given \(i\), the \(i^{th}\) sub-interval given by \([\frac{ad^i}{c^i}, \frac{ad^{i+1}}{c^{i+1}}]\) maps into the interval \([c, d]\) for \(s = s_i\) (where \(s_i = \frac{c}{a} (\frac{c}{d})^i\)).

Hence, we have shown that each element of \([a, b]\) belongs to some interval \(I\) in \(C\). Also, for every interval \(I\) in \(C\), there exists an element \(s \in S\) that maps \(I\) to \([c, d]\).

The size \(m\) of \(S\) is such that \(b \leq a \left(\frac{d}{c}\right)^m\). Hence, for \(m \geq \frac{\log b}{\log d}\), the lemma holds true.

We apply the lemma for parameters \(a = \frac{2}{8}, b = \frac{8}{8}, c = \frac{5}{8}\) and \(d = \frac{7}{8}\). This shows that for every element \(e \in [a, b]\) the set \(\left\{\frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}\right\}\) contains an element \(\alpha\) such that \(\alpha e \in [c, d]\). We shall use this lemma in the proof of the next theorem. The next theorem proves the correctness of the algorithm and also bounds the approximation ratio obtained.

**Theorem 4.3.** The set of selected points \(S \in I\). Also, \(|S| \in \Omega\left(\frac{s}{\log n}\right)\), with probability \(\geq \frac{1}{8 \log \log n}\), where \(s\) is the size of the maximum independent set in \(I\).

Proof. \(S \in I\) follows directly from the fact that we only add an point \(v\) to \(S\) if \(v\) is connected to a specific right block \(r\) (obtained in step 4). Hence all elements in \(S\) share \(r\) as a neighbor and are independent.

We now prove the bound on the size of \(S\). Let \(r\) be a block in \(R\). Denote the neighbors of \(r\) by \(N(r)\). For a given first half \(F\) generated by a permutation \(\sigma\), we say that a right block \(r\) is badly sampled if \(|N(r) \cap F| \geq \frac{3}{4}|N(r)|\) or \(|N(r) \cap F| \leq \frac{1}{4}|N(r)|\). If \(r\) is not badly sampled we call it well-sampled. If the algorithm selects a well-sampled block \(r\) (in step 4), then \(|S| \geq \frac{N(r)}{4}\).

We shall now prove that the algorithm picks a block of degree at least \(\frac{s}{256 \log n}\) with probability at least \(\frac{1}{8 \log \log n}\). Lemma 4.4 proves that any particular right block is well-sampled with high probability.
Lemma 4.4. For a block $r \in R$ the probability that it is badly sampled is $\leq 2^{-\frac{|N(r)|}{16}}$.

Proof. Let $N(r) = \{v_1, \ldots, v_k \in L\}$. Since elements of $L$ are seen in random order, the probability that a particular block $v_i \in F = \frac{1}{2}$.

Define a random variable $X_r = |N(r) \cap F|$ which depends on the random sample $F$. $E[X_r] = \frac{1}{2}|N(r)|$. Using Chernoff Bounds (for example Chung and Lu [2006]), we see that $Pr[|X_r - E[X_r]| \geq \lambda] \leq 2e^{-\frac{\lambda^2}{2(|N(r)| + \lambda/3)}} \leq 2e^{-\frac{\lambda^2}{N(r) + \frac{\lambda}{3}}}$.

Choose $\lambda = \frac{1}{4}|N(r)|$. $Pr[|X_r - E[X_r]| \geq \frac{1}{4}|N(r)|] \leq 2^{-\frac{|N(r)|^2/16}{N(r) + |N(r)|}} < 2^{-\frac{|N(r)|}{16}}$. 

\[ \begin{array}{c|c|c|c|c|}
 s & s/2 & s/4 & s/8 \\
\hline
 B_1 & B_2 & B_3 \\
\end{array} \]

Figure 4.3: Bucketing the independent sets

The algorithm needs to make sure that it selects a high degree block. This is achieved by selecting a large threshold in step 3. Applying Lemma 4.4 to the highest degree block gives that it is well-sampled with high probability. Since $s$ is the highest degree, the block will have at least $\frac{s}{4}$ points in $F$ with high probability. Hence, the highest degree block in $F$ has degree $\hat{s} \in [s, s/4]$ with high probability. Lemma 4.2 shows that there exists an $\alpha \in \left\{ \frac{5}{2}, \frac{5}{7}, \frac{5}{2}, \frac{5}{7}, \frac{5}{2}, \frac{5}{7}, \frac{5}{2}, \frac{5}{7} \right\}$ such that $t = \alpha \hat{s}/2^i \in \left[ \left( \frac{3}{4} - \frac{1}{8} \right) \frac{s}{2^i}, \left( \frac{3}{4} + \frac{1}{8} \right) \frac{s}{2^i} \right]$. This happens with probability $\frac{1}{4}$ since $\alpha$ is chosen uniformly at random. We classify right blocks into bins $B_1, \ldots, B_{256\log n}$ according to their degree in the original hypergraph. $B_i$ contains all blocks $r \in R$ which have degree $d \in \left[ \frac{s}{2^i-1}, \frac{s}{2^i} \right]$.

We shall now prove that the oracle actually picks a well-sampled block. While any
particular block is well-sampled with high probability (Lemma 4.4), the number of blocks is large. Hence, we cannot directly apply the union bound. We divide the proof into two cases:

Case 1: \(|B_1| + |B_2| < 2^{s_{128}}\)

The algorithm chooses \(j = 1\) with probability \(\frac{1}{8 \log \log n}\). Hence with probability \(\frac{1}{32 \log \log n}, t \in \left[\frac{5}{8}s, \frac{7}{8}s\right]\). None of the blocks in bins \(B_{>2}\) will be selected since they cannot pass this threshold. Let \(q\) be a block in \(B_1 \cup B_2\). By lemma 4.4, the probability that \(q\) is badly sampled \(\leq 2^{-|N(q)|/16} \leq 2^{-s/64}\). Using the union bound, the probability that there is a block in \(B_1 \cup B_2\) that is badly sampled \(\leq 2^{s/128} \cdot 2^{-s/64} = 2^{-s/128}\).

This means the algorithm selects a well-sampled block \(r\) of degree at least \(s/4\) and will select at least \(s/16\) elements with probability \(\frac{1}{32 \log \log n}\). This gives us a \(O(\log \log n)\) approximation.

Case 2: \(|B_1| + |B_2| \geq 2^{s_{128}}\)

This is the harder case. We start with a lemma that states the existence of consecutive bins, such that the bins with the larger sets contain more blocks than the bins with the smaller sets.

**Lemma 4.5.** There exists and index \(l \in [0, 8 \lceil \log \log n \rceil]\) such that \(|B_l| + |B_{l+1}| \geq |B_{l+2}| + |B_{l+3}|\).

**Proof.** Recall \(\binom{n}{k} \leq n^k\). For \(k = \frac{s}{256 \log n}\), the equation becomes:

\[
\left( \frac{n}{256 \log n} \right)^{\frac{s}{256 \log n}} \leq n^{\frac{s}{256 \log n}} = \left( n^{\frac{1}{256 \log n}} \right)^{\frac{s}{256}} = 2^{\frac{s}{256}}
\]

\(|B_{8\lceil \log \log n \rceil}| + |B_{8\lceil \log \log n \rceil + 1}| \leq 2 \left( \frac{n}{256 \log n} \right)^{\frac{s}{256}} \leq 2^{-2}\leq |B_1| + |B_2|\). Hence, there exists an index \(l \in [0, 8 \lceil \log \log n \rceil]\) such that \(|B_l| + |B_{l+1}| \geq |B_{l+2}| + |B_{l+3}|\). \(\square\)

Consider the event \(t \in \left[\frac{5}{8}s, \frac{7}{8}s\right] \cup \left[\frac{5}{8}\frac{s}{2^{l+3}}, \frac{7}{8}\frac{s}{2^{l+3}}\right]\). It occurs if the algorithm chooses \(i = l + 3\).
(in step 3) and $\alpha$ such that $\alpha \frac{\hat{s}}{s} \in \left[ \frac{5}{8}, \frac{7}{8} \right]$. Lemma 4.2 shows that there always exists such an $\alpha$. The correct $\alpha$ is chosen with probability at least $\frac{1}{4}$. This happens with probability at least $\frac{1}{8 \log \log n} \times \frac{1}{4} = \frac{1}{32 \log \log n}$.

Let $Y_i$ be the random variable that denotes the number of badly sampled blocks in bucket $B_i$. Lemma 4.4 says that any given block in $B_{\leq i}$ is well-sampled with probability at least $1 - 2^{-\frac{\hat{s}}{s}}$ i.e. $\mathbb{E}[Y_i] \leq |B_i|2^{-\frac{\hat{s}}{s}}$. By Markov’s Inequality, $P_r \left[ Y_i \geq \frac{|B_i|}{4} \right] \leq 4 \cdot 2^{-\frac{\hat{s}}{s}}$.

We show that the ratio of well-sampled blocks to badly sampled blocks is large. None of the blocks in bins $B_{i+3}$ can pass the threshold. Even if all the blocks in bins $B_{i+2}$ and $B_{i+3}$ are badly sampled, with probability at least $1 - 4 \cdot 2^{-\frac{\hat{s}}{s}}$, the total number of badly sampled blocks $\leq \sum_{m \leq i} \frac{|B_m|}{4} + B_{i+2} + B_{i+3} \leq \sum_{m \leq i+1} \frac{|B_m|}{4} + B_i + B_{i+1} \leq \frac{5}{4} \sum_{m \leq i+1} |B_m| = \frac{5}{3} \sum_{m \leq i+1} \frac{3}{4} |B_m| \leq \frac{5}{3} \mathbb{E}[ \ # \text{ of well-sampled vertices}]$. Hence, the oracle chooses a well-sampled block with probability at least $\frac{3}{8}$.

Also, $i < 8 \lceil \log \log n \rceil$ implies the size of any block in bin $B_i$ is at least $\frac{s}{256 \log n}$. Hence $|S| \in \Omega \left( \frac{s}{\log n \log \log n} \right)$ and we get a $O(\log n \log \log n)$ approximation. \qed

This completes the proof of the theorem.

### 4.4 Weighted Elements

Until now, we assumed that the elements did not have weights. Now we shall see an algorithm when each element can have an unknown weight, that we get to know when we actually see the element.

Let $W$ be the largest weight among all elements in $V$. We split elements into $\log n$ partition depending on their weight. Elements of weight $w \in \left[ \frac{W}{2^j}, \frac{W}{2^{j+1}} \right]$ fall in partition $j$. We ignore all elements of weight $< \frac{W}{n}$, since they collectively have weight smaller than $W$. Hence, we have $\log n$ partitions. Consider the partition which has the
largest weight independent set.

We add a preprocessing step to the algorithm presented in the previous section. The preprocessing algorithm selects a partition uniformly at random from all partitions and selects elements only from this partition. The probability that we select the heaviest partition is $\frac{1}{\log n}$.

Note that if we have an $\alpha$ approximation algorithm for selecting the largest independent set for elements in the partition with the largest size, then the algorithm with the preprocessing step gives a $\alpha \log n$ approximation algorithm for the original independent set. But all elements in a partition have weight within a factor 2 of each other. Hence, if we ignore all the weights of elements in a partition, then we lose at most a factor 2 in the approximation guarantee. Coupled with the algorithm in the previous section, this gives a $O\left(\log^2 n \log \log n\right)$ approximation algorithm for the general problem where elements have weights.

### 4.5 Future direction using the Oracle $O_2$

We shall now discuss the problem of selecting a maximum independent set of unweighted vertices, given only the weaker oracle $O_2$. Recall that given two sets of elements $S, T$ such that $S \subseteq T$ and $S \in \mathcal{I}$, the oracle $O_2$ returns a maximum independent subset $W$ such that $S \subseteq W \subseteq T$. $O_2$ ensures that we select all the elements of $S$ and among all independent subsets of $T$ which also include $S$, we choose the largest one. If there are multiple maximum independent subsets, it arbitrarily chooses one. We do not have concrete results in this case and so we sketch two directions for further research that seem fruitful in sections 4.5.1 and 4.5.2.

#### 4.5.1 Approximating the offline oracle

Let $\mathcal{H} = (V, \mathcal{I})$ be a hypergraph. Consider the counting version of the offline oracle function $O_2$, given by $O_{\text{offline}} : 2^V \times 2^V \to \mathbb{N}$. Given two sets, green and red,
$O_{\text{offline}}(G, R)$ returns the maximum number of elements of $V \setminus R$ that can be added to $G$ such that the resulting set is independent.

Hypothetically, suppose that the online algorithm has access to the offline oracle $O_{\text{offline}}$. We could use this to select a large independent set: Set $G = R = \emptyset$. $O_{\text{offline}}(\emptyset, \emptyset)$ tells us the size of the maximum independent set $s$. Look at elements one by one. If the element $x$ is such that $O_{\text{offline}}(G \cup x, R) = s$, add $x$ to $G$ else add it to $R$. Clearly, this chooses a maximum independent set. Of course, since we want to construct an online algorithm, the oracle $O$ does not have access to elements that have not arrived yet. The approach we take is to approximate the behavior of the offline oracle $O_{\text{offline}}$. We now give a candidate oracle that does not have access to future elements, but that approximates the offline version of the oracle $O_{\text{offline}}$.

The Mean Oracle

Let $V$ be all the elements in the hypergraph. Consider a function $f_S$ of the following type: Given sets green and red, $(G, R)$ and a fixed subset $S \subseteq V - (G \cup R)$, the function $f_S$ outputs the size of the largest set $I \subseteq S$ such that $I \cup G$ is independent and $I \cap R = \emptyset$.

We define the mean oracle as the weighted average over all sets $S$, of the size output by the above function $f_S$. We define the mean oracle as follows

$$f_{\text{mean}}(G, R) = \sum_{S \subseteq V - (G \cup R)} 2^{-|S|} f_S(G, R)$$

We call this the mean oracle since it gives, for any two sets $G$ of selected elements and $R$ of rejected elements, the mean number of remaining elements that can be added to $G$, such that the resulting set is independent. We shall now define a smoothness property that we want the mean oracle to follow.

1) $f_{\text{mean}}(G \cup x, R) \geq f_{\text{mean}}(G, R) - \log n$

2) $f_{\text{mean}}(G, R \cup x) \geq f_{\text{mean}}(G, R) - \frac{1}{n}$
The initial value of the mean oracle is at least \( s \), the size of the largest independent set.

Suppose we can show that \( \forall x \in V - (G \cup R) \), either 1 or 2 holds. (It's OK if both hold too). For each new element, the smoothness properties ensure that either we increase the size of the selected independent set (if option (1) holds), or if we reject the element, then there are still enough elements left that we can use to extend the current set of selected elements. That would give us a way to construct an online algorithm which uses this mean oracle and gives a log \( n \) competitive ratio.

### 4.5.2 A labeling game

In this section, we consider a general strategy to solve the secretary problem. Given a set \( R \) and an independent set \( S \), let \( N(R, S) = \) Number of independent sets in \( R \) that contain \( S \). Consider the following strategy:

1. Look at half the elements in random order. Don’t select any element. Call these elements red. The remaining are blue. Calculate \( N(R, \phi) = \) Number of independent sets in \( R \).

2. Set \( S \leftarrow \phi \)

3. For each element \( e \), calculate \( p_e = \frac{N(R, S \cup e)}{N(R, S)} \). Accept \( e \) with probability \( p_e \) and set \( S \leftarrow S \cup e \).

We want to show that this strategy does actually give us a log \( n \) approximation algorithm. We start by stating some lemmas.

**Lemma 4.6.** If we randomly color each element \( R \) or \( B \) with probability \( \frac{1}{2} \), then \( N(R) \geq 2^\frac{s}{100} \) with probability \( \geq \frac{1}{2} \).

**Lemma 4.7.** We do not accept any element \( e \) for which \( p_e < \frac{1}{n^2} \), with probability \( \geq \left( 1 - \frac{1}{n} \right) \).
Proof. Since there are \( n \) elements, using union bound, we get that the probability that we do select some element for which \( p_e \leq \frac{1}{n^2} \) is \( \leq \sum_{\{p_e < \frac{1}{n^2}\}} p_e < \frac{1}{n} \).

Hence, if \( S \) is the set we have accepted so far, then \( \forall i : \frac{N(R, S_i)}{N(R, S_{i-1})} \geq \frac{1}{n^2} \).

**Lemma 4.8.** The number of accepting steps before \( N(R, S) \) becomes smaller than \( 2^{\frac{s}{200}} \) is at least \( \frac{s}{400 \log n} \).

*Proof.* At each step,

\[
N(R, S_k) = \frac{N(R, \phi) \cdot N(R, S_1) \cdot \ldots \cdot N(R, S_k)}{N(R, S_{k-1})} \geq 2^{s/100} \frac{1}{n^{2k}} \]

\[
= 2^{s/100 - 2k \log n}
\]

For \( k \leq \frac{s}{400 \log n} \), \( N(R, S_k) \geq 2^{s/100 - s/200} \geq 2^{s/200} \)

To show that the algorithm actually works as intended, we need another lemma which proves that we can continue accepting elements with high probability. The proof of this rests on the following equation

\[
\sum_{e \in B} N(R, S_i \cup e) \geq \frac{s}{4 \log n} N(R, S_i)
\]  \hspace{1cm} (4.1)

**Conjecture 4.9.** We accept an element at each step \( i \) with high probability

*Sketch.* At step \( i \), we know that \( N(R, S_i) \geq 2^{s/100 - i \log n} \). Suppose we can show that the remaining elements also satisfy the property that \( N(B, S_i) \geq 2^{s/100 - i \log n} \). Then, let \( p_e = \frac{N(B, S_i \cup e)}{N(B, S_i)} \). The probability that we accept no element is equal to \( p = \prod_{e \in B} (1 - p_e) \leq \exp \left(- \sum_{e \in B} p_e \right) = \exp \left(- \frac{\sum_{e \in B} N(R, S_i \cup e)}{N(R, S_i)} \right) \leq \exp \left(- \frac{s}{4 \log n} \right) \)

The last deduction hinges on the assumption that we can prove 4.1

To prove equation 4.1, we construct the following game.
The Game

Consider the following single player game:

1. Start with \( n \) elements. Each element has weight 1. Color all the elements blue.

2. Repeat the next two steps \( \frac{n}{2} \) times.

3. An element \( e \) is chosen, uniformly at random, from the set of blue elements. Color it red. Fix its label to its current weight \( w(e) \).

4. The player can now select any set of blue elements \( B \) and increase their weight.

We imagine an adversary playing this game, and use it to model the moves made by the adversary. At the end of the game, each element has a label, giving its weight. We want to show that the adversary cannot make the weight of red elements much larger than those of the blue elements. In particular we want to prove that, not matter what strategy the adversary uses,

\[
S = \sum_{e \in B} w(e) \geq \frac{3}{4} \sum_{e \in R} w(e) \text{ with probability } \geq (1 - 2^{-s/10}) \tag{4.2}
\]

The correspondence of the game to the algorithm given in the previous section is as follow: The weight of an element corresponds to the number of sets it belongs to. The random move corresponds to a new element arriving. And selecting an element corresponds to adding the element to the set of selected elements. The player in the game corresponds to the structure of the hypergraph. This also now gives us ways to constrain the move the player can make.

Equation 4.2 is not true unless we impose some constraints on the adversary. What are the minimal conditions that we need to impose such that the above assertion becomes true? Here are two natural conditions that we can impose on the adversary. We get these from the motivation that this game reflects selecting elements in an independence set system.
1. The weight of elements in \( B \) cannot be more than doubled.

2. If the element \( e' \) is selected in Step 3, then the weight of each \( e \in B \) can go up by at most \( w(e') \).

If we use the game to prove that the fraction of blue elements that can be added to the selected elements is at least \( \frac{1}{\log n} \) i.e. \( \sum_{e \in B} \frac{N(R,e)}{N(R,\phi)} \geq \frac{s}{2 \log n} \), then this will give us the online approximation algorithm.

Below we discuss some example strategies that the adversary can use:

1. Serial Increase: Keep one element in mind and keep increasing its weight by 1, till it is selected. Once it is selected, choose another element and keep doing this. This does not seem too good, but it leads to the next strategy.

2. Parallel Increase: Keep a set \( T \) in mind. Whenever any element is selected by the algorithm, increase the weight of all elements in \( T \) by the maximum that is allowed by the constraints. Note that the adversary wins with probability \( 2^{-T} \).

Is this the best the adversary can do, or is there a better strategy?

### 4.6 Conclusion

In this chapter, we looked at the secretary problem for general independence set systems. We saw that there was a lower bound of \( \Omega \left( \frac{\log n}{\log \log n} \right) \) on the competitive ratio of any online algorithm. In Section 4.3, using a strong random sampling oracle, we gave a \( O(\log n \log \log n) \) approximation algorithm for unweighted elements. We also saw a \( O(\log^2 n \log \log n) \) approximation for weighted elements in Section 4.4. While we leave the problem of solving the problem using the weaker oracle open, we looked some directions for further research in Section 4.5.
Differential Privacy
Chapter 5

Differential Privacy

5.1 Introduction

Privacy Mechanisms: Agencies such as medical establishments, survey agencies, governments use and publish aggregate statistics about individuals; this can have privacy implications. Consider the query: *Q: How many adults from San Diego contracted the flu this October?* The government can use the query result to track the spread of flu, and drug companies can use it to plan production of vaccines. However, knowledge that a specific person contracted the flu could be used to deny her health insurance based on the rationale that she is susceptible to disease. As discussed by Sweeney [2002], and as is exemplified by the AOL search data scandal\(^1\) and the breach of the Netflix challenge dataset [Narayanan and Shmatikov, 2008], seemingly benign data publications can have privacy implications. Thus, it is important to think rigorously about privacy. The framework of differential privacy [Dwork, 2006] does this, and is applicable widely (see Section 5.2.8).

Mechanisms guarantee differential privacy by perturbing results – they add random noise to the query result, and guarantee protection against all attackers, whatever their side-information or intent (see Section 5.2.1 for a formal definition).

Our Utility Model: The addition of noise increases privacy but intuitively reduces utility of the query result. To understand this privacy-utility trade-off, we propose a formal decision-theoretic model of utility. Decision-theory is a widely applied field that

\(^1\)http://en.wikipedia.org/wiki/AOL_search_data_scandal
provides mathematical foundations for dealing with preferences under uncertainty. The use of decision theory in this context is appropriate because, as we discussed above, mechanisms guarantee differential privacy by introducing uncertainty.

In our model of utility (see Section 5.2.3 for details), the user of information, i.e. the information consumer has side-information – for instance, knowledge of the population of San Diego is an upper bound on the result of the query $Q$. It has a loss-function that expresses it’s tolerance to inaccuracy. It is rational in the sense that it combines information from the mechanism with its side-information optimally with respect to its personal loss-function. It is risk-averse in the sense that it would like to minimize worst-case loss over all scenarios.\(^2\)

Given the privacy parameter, the loss-function and the side-information of an information consumer it is possible to identify an optimal mechanism – a mechanism that is differentially private and that maximizes its utility. See Section 5.2.4 for an algorithm to find such a mechanism.

**Non-Interactive Settings:** Very often aggregate statistics, like answers to $Q$, are published in mass media as opposed to following a query-response form (e.g. the California Department of Public Health, H1N1 Flu–Data Tables \(^3\)). In such cases neither the information consumer nor it’s loss-function and side-information are known in advance. Thus it seems hard to identify the optimal mechanism for a information consumer.

Nevertheless, we show that it is possible to deploy an optimal mechanism without knowledge of the information consumer’s parameters. Furthermore, this mechanism is universally optimal for all information consumers, no matter what their side-information or loss-function.

\(^2\)Ghosh, Roughgarden, and Sundararajan [2009] propose a model with most of these features, but assumes that information consumers are Bayesian and have a prior over the query-result.

\(^3\)http://www.cdph.ca.gov/HealthInfo/discond/Documents/H1N1-Data-Table-CA-Cases-by-County-102409.pdf
How can we identify the optimal mechanism without knowledge of the information consumer’s parameters? The apparent paradox is resolved by relying on the information consumers’ rationality, i.e., each information consumer uses its personal loss-function and side-information to actively transform the output of the deployed mechanism. For a certain class of queries called count queries, when the deployed mechanism is a certain geometric mechanism, this transformation is effective enough to result in the optimal mechanism for the information consumer—a fact that we will establish via linear-algebraic proof techniques.

**Multiple Levels of Privacy:** We also show how to simultaneously release the query result at different levels of privacy to different information consumers. This is useful, for instance, when we want to construct two versions of the report on flu statistics, one which prioritizes utility for the eyes of government executives, and a publicly available Internet version that prioritizes privacy.

A naive solution is to perturb the query results differently, independently adding differing amounts of noise each time. The drawback is that consumers at different levels of privacy can collude and combine their results to cancel the noise (as in Chernoff bounds). An alternate way is to correlate the noise added to different outputs. We give an algorithm to achieve this that makes the data release collusion-resistant.

In this chapter we focus on a single query; the complexity comes from a rich model of consumer preferences, where we consider different utility functions for each consumer and optimize for each of them. Blum, Ligett, and Roth [2008], Hay, Rastogi, Miklau, and Suciu [2010], Hardt and Talwar [2010] exploit similarities between the queries to obtain extension to multiple queries with good utility guarantees. However, they do not consider a rich consumer preference model. Our results could be used as a building block while answering multiple queries.
5.2 Model and Results

We gave a informal description of our model and results in the Introduction. In this section, we formally define our model and discuss the main results. The proofs of the results are presented in Sections 5.3, 5.4.

5.2.1 Privacy Mechanisms and Differential Privacy

A database is a collection of rows, one per individual. Each row is drawn from an arbitrary domain $D$; for instance, in our running example, a row of the database has the name, age, address and medical records of a single individual. A database with $n$ rows is thus drawn from the domain $D^n$.

We will focus on a class of queries, called count queries, that frequently occur in surveys: Given a predicate $p : D \rightarrow \{\text{True, False}\}$, the result of a count query is the number of rows that satisfy this predicate, a number between 0 and the database size, $n$. $Q$ is an example of a count query with the predicate: individual is an adult residing in San Diego, who contracted flu this October. Though simple in form, count queries are expressive because varying the predicate naturally yields a rich space of queries.

We guarantee differential privacy to protect information of individual database participants. Differential privacy is a standard, well-accepted definition of privacy [Dwork, 2006] that has been applied to query privacy [Dwork and Nissim, 2004, Dwork, McSherry, Nissim, and Smith, 2006, Nissim, Raskhodnikova, and Smith, 2007], privacy preserving machine learning [Blum, Ligett, and Roth, 2008, Kasiviswanathan, Lee, Nissim, Raskhodnikova, and Smith, 2008] and economic mechanism design [McSherry and Talwar, 2007]. A fixed count query maps the database $d$ to a number which belongs to the set $N$. A privacy mechanism $M$ for a fixed count query is a probabilistic function that maps a database $d \in D^n$ to the elements of the set $N = \{0 \ldots n\}$. These can be represented, for each $d \in D^n$, by $\{m_{d,r}\}_{r \in N}$, which gives for each database $d \in D^n$ the
probability that $M$ outputs $r$. For the database $d$, the mechanism releases a perturbed result by sampling from the distribution $\{m_{d,r}\}_{r \in \mathbb{N}}$.

The Geometric Mechanism [Ghosh, Roughgarden, and Sundararajan, 2009] is a simple example of a privacy mechanism. It is a discrete version of the Laplace Mechanism from [Dwork, McSherry, Nissim, and Smith, 2006].

**Definition 5.1** ($\alpha$-Geometric Mechanism). When the true query result is $f(d)$, the mechanism outputs $f(d) + Z$. $Z$ is a random variable distributed as a two-sided geometric distribution: $Pr[Z = z] = \frac{1 - \alpha}{1 + \alpha} \alpha^{|z|}$ for every integer $z$.

![Probability distribution](image)

**Figure 5.1:** The probability distribution on outputs given by the Geometric Mechanism for $\alpha = 0.2$ and query result $5$.

Informally, a mechanism satisfies differential privacy if it induces similar output distributions for every two databases that differ only in a single individual’s data, thereby ensuring that the output is not sensitive to any one individual’s data\(^4\). Formally, differential privacy is defined as follows [Dwork, McSherry, Nissim, and Smith, 2006]:

\(^4\)Thus any attack on an individual’s privacy that can be constructed using the perturbed query result with this individual present in the database can also be constructed, with a similar success rate, without this individual present in the database. See See [Dwork, McSherry, Nissim, and Smith, 2006, Kasiviswanathan and Smith, 2008] for details of such semantics of differential privacy.
Given a privacy parameter \( \alpha \in [0, 1] \) and two database \( d_1, d_2 \in D^n \) that differ in at most one individual’s data, a mechanism \( M \) is \( \alpha \)-differentially private, if for all elements \( r \) in the range of the mechanism:

\[
\frac{1}{\alpha} x_{d_1,r} \geq x_{d_2,r} \geq \alpha x_{d_1,r}.
\]

The parameter \( \alpha \) can be varied in the interval \([0, 1]\) to vary the strength of the privacy guarantee—when \( \alpha = 0 \), the above definition is vacuous and there is no privacy, whereas when \( \alpha = 1 \), we effectively insist on absolute privacy— the query result cannot depend on the database because we require distributions over perturbed results to be identical for neighboring databases.

### 5.2.2 Oblivious Mechanisms

We will focus in this chapter on a class of privacy mechanisms that are oblivious. A mechanism is *oblivious* if it sets up an identical distribution over outputs for every two databases that have the same unperturbed query result. Naturally, an implementation of an oblivious mechanism need only have access to the true query result—the *input*—and can be oblivious to the database itself. An oblivious mechanism for count queries can be expressed by the set of probability masses for every \( i \in N : \{x_{i,r}\}_{r \in N} \), where \( x_{i,r} \) is the probability that the mechanism outputs \( r \) when the true result is \( i \). Appendix 5.5 shows that this restriction to oblivious mechanisms is without loss of generality. The geometric mechanism (Definition 5.1) only depends on the query result \( f(d) \) and not on the database \( d \) itself; so it is a oblivious mechanism.

The query result for a count query can change by at most one when we change any one row of the database, so we can rewrite the definition of differential privacy as follows:

**Definition 5.2** (Differential Privacy for Count Queries). An oblivious mechanism for count queries for \( \alpha \in [0, 1] \) is \( \alpha \)-differentially private if for all \( i \in \{0 \ldots n - 1\}, r \in N : \)

\[
\frac{1}{\alpha} x_{i,r} \geq x_{i+1,r} \geq \alpha x_{i,r}.
\]
Observe that the geometric mechanism is \( \alpha \)-differentially private because for two adjacent inputs \( i, i + 1 \in N \), and any output \( r \in N \), \( \frac{x_{i,r}}{x_{i+1,r}} \in [\alpha, 1/\alpha] \).

### 5.2.3 Minimax Information Consumers

We now discuss our model of an information consumer’s utility. The loss-function \( l(i, r) : N \times N \to \mathbb{R} \) specifies the loss of the information consumer, given the mechanism outputs \( r \) when the true result is \( i \). We only assume that the loss-function is monotone non-decreasing in \( |i - r| \), for every \( i \). That is, the consumer becomes unhappier as the returned answer is further away from the true result.

Consider some examples of valid loss-functions: The loss-function \( l(i, r) = |i - r| \) quantifies the mean error—for our query \( Q \), this loss-function may be a reasonable one for the government who want to keep track of the rise of flu. The loss-function \( l(i, r) = (i - r)^2 \) quantifies the variance in the error—this may be reasonable for a drug company who wants to ensure that they don’t over-produce or under-produce the flu drug. The loss-function \( l(i, r) = \begin{cases} 0 & \text{if } i = r \\ 1 & \text{if } i \neq r \end{cases} \), measures the frequency of error.

Additionally, we will assume that the information consumer has side information \( S \subseteq N \), i.e., the information consumer knows that the query result cannot fall outside the set \( S \). For instance, knowledge of the population of San Diego yields an upper-bound on the query result. The drug company may also know how many people bought its flu drug this month, yielding a lower bound on the query result.

For any specific input \( i \), the loss-function \( l \) allows us to evaluate the information consumer’s dis-utility as the expected loss over the coin tosses of the mechanism: \[ \sum_{r \in N} l(i, r) \cdot x_{i,r}. \] To quantify the overall loss, we follow the minimax decision rule, i.e., we take the worst-case loss over all inputs in the set \( S \) [Loomes and Sugden, 1982]. This amounts to the information consumers being risk-averse. Hence, the dis-utility of
the mechanism $x$ to the consumer $c$ is:

$$L(x) = \max_{i \in S} \sum_{r \in N} l(i, r) \cdot x_{i,r}$$

(5.1)

5.2.4 Interactions of Information Consumers with Mechanisms

As mentioned in the Introduction, information consumers actively interact with the mechanism to induce a new mechanism; we now discuss the mechanics of this interaction.

Motivation

The following example argues why a rational information consumer will not accept the mechanism’s output at face value.

**Example 1.** Recall the query $Q$ defined in the Introduction. Suppose that the information consumer is a drug company, who knows that $l$ individuals in San Diego bought its flu drug in the month of October. Thus the query result $Q$ must be at least $l$; the information consumer cannot conclude that the query result is exactly $l$ because some individuals with flu may have bought a competitor's drug, or bought no drug at all. Thus it has side-information $S = \{1 \ldots n\}$.

Suppose we deploy the geometric mechanism for the query $Q$. This mechanism returns with non-zero probability outputs outside the set $\{1 \ldots n\}$. Such outputs are evidently incorrect to the information consumer, and naturally it makes sense for the information consumer to map these results within the set $\{1 \ldots n\}$. Though it is not clear what the best way of doing so is, a reasonable rule may be to re-interpret results less than $l$ as $l$, and results larger than $n$ as $n$.

Feasible Interactions

Before we discuss the optimal way for an information consumer to interact with the mechanism, we describe the space of feasible interactions. On receiving a query result
r from the mechanism, the consumer can reinterpret it as a different output. This reinterpretation can be probabilistic and can be represented by a set of probability masses \( \{T_{r,r'} : r' \in N \} \) which gives for each result \( r \), the probability that the consumer will reinterpret it as the output \( r' \). Such an interaction induces a new mechanism for the user. Suppose the deployed mechanism is represented by the set of probability masses \( \{y_{i,r} : i, r \in N \} \), and the induced mechanism as the probability masses \( \{x_{i,r'} : i, r \in N \} \), then \( x_{i,r'} = \sum_{r \in N} y_{i,r} \cdot T_{r,r'} \). We formalize this in a definition.

**Definition 5.3 (Derivability).** Given two mechanism \( x \) and \( y \), we say that mechanism \( x \) can be derived from \( y \) if and only if, for every \( r \in N \), there exists a set of probability masses \( \{T_{r,r'} : r' \in N \} \) such that for every \( i, r' \in N \): \( x_{i,r'} = \sum_{r \in N} y_{i,r} \cdot T_{r,r'} \).

**Optimal Interactions**

Given a deployed mechanism \( y \), the optimal interaction \( T^* \) is one that minimizes the information consumer’s maximum loss on the induced mechanism. The optimal interaction can be computed using a simple linear program. There are \( n^2 \) variables: one for each \( T^*_{r,r'} \in N \). The objective function is obtained by minimizing the loss to the consumer if it uses interaction \( T^* \). The constraints are obtained from the fact for each \( r \), the entries \( T^*_{r,r'} \) form a probability distribution and hence sum up to 1 and that all entries of \( T^* \) are positive. The actual linear program is given as:

\[
\begin{align*}
\text{minimize} & \quad \max_{i \in S} \sum_{r \in N} x_{i,r} \cdot l(i,r) \\
x_{i,r} & = \sum_{r \in N} y_{i,r} \cdot T^*_{r,r'} \quad \forall i \in N, \forall r \in N \\
\sum_{r' \in N} T^*_{r,r'} & = 1 \quad \forall r \in N \\
T^*_{r,r'} & \geq 0 \quad \forall r \in N, \forall r' \in N
\end{align*}
\]
5.2.5 Optimal Mechanism for a Single Known Information Consumer

Identifying the optimal mechanism for a specific consumer reduces to the following:
Identify a level of privacy $\alpha$ with which to release the result. Find the consumer’s
loss-function and side-information. Identify an $\alpha$-differentially private mechanism such
that the mechanism induced by the consumer’s optimal interaction (as described in the
previous section), has the best possible utility.

In the case of a single information consumer, we can obviate the need for the infor-
mation consumer to reinterpret the deployed mechanism’s output: Suppose there is a
mechanism $y$ with post-processing $T$ that induces a mechanism $x$. Clearly, presenting
$x$ directly to the information consumer yields at least as much utility for it. All we have
to ensure is that $x$ is $\alpha$-differentially private, and a simple proof (omitted) shows that
this is indeed so.

Thus, to identify the optimal mechanism for a specific information user, it suffices
to search over $\alpha$-differential mechanisms. For a given consumer $c$ with loss-function

$$L(l, S) = \max_{i \in S} \sum_{r \in N} x_{i,r} \cdot l(i, r)$$

and privacy parameter $\alpha$, the optimal differentially private mechanism $M_c$ is the solution
to a simple linear program. Like in the previous section, there are $n^2$ variables one for
each matrix entry of the mechanism $x$. The objective is to minimize the user’s loss
function. The constraints are obtained by the facts that

1. $x$ is differentially private. So the variables $x_{i,r}$ must satisfy Definition 5.2

2. For each input $i$, elements $x_{i,r}$ form a probability distribution and hence sum up
to 1.

3. All elements $x_{i,r}$ are positive

Writing this as an optimization problem we get:
minimize \( \max_{i \in S} \sum_{r \in N} x_{i,r} \cdot l(i,r) \)

\[
\begin{align*}
x_{i,r} - \alpha \cdot x_{i+1,r} & \geq 0 & \forall i \in N \setminus \{n\}, \forall r \in N \\
\alpha \cdot x_{i,r} - x_{i+1,r} & \leq 0 & \forall i \in N \setminus \{n\}, \forall r \in N \\
\sum_{r \in N} x_{i,r} & = 1 & \forall i \in N \\
x_{i,r} & \geq 0 & \forall i \in N, \forall r \in N
\end{align*}
\]

We can convert it into a Linear Program, the solution of which gives us \( x^* \).

minimize \( d \)

\[
\begin{align*}
d - \sum_{r \in N} x_{i,r} \cdot l(i,r) & \geq 0 & \forall i \in S \\
x_{i,r} - \alpha \cdot x_{i+1,r} & \geq 0 & \forall i \in N \setminus \{n\}, \forall r \in N \\
\alpha \cdot x_{i,r} - x_{i+1,r} & \leq 0 & \forall i \in N \setminus \{n\}, \forall r \in N \\
\sum_{r \in N} x_{i,r} & = 1 & \forall i \in N \\
x_{i,r} & \geq 0 & \forall i \in N, \forall r \in N
\end{align*}
\]

To deploy this mechanism \( x^* \), we first compute the true query result, say \( i \), then sample the perturbed result \( r \) from the distribution \( \{x^*_{i,r} : \forall r \in N\} \), and release the sampled result. Table 5.1(a) gives an example of a optimal mechanism for a particular information consumer.

5.2.6 Optimal Mechanism for Multiple Unknown Information Consumers

How can we extend the results of the previous section to multiple consumers? The naive solution is to identify and separately deploy the optimal mechanism for each information consumer as described in the previous section.
\[
\begin{bmatrix}
\frac{2}{3} & \frac{5}{17} & \frac{1}{25} & \frac{1}{98} \\
\frac{1}{6} & \frac{7}{11} & \frac{7}{44} & \frac{2}{49} \\
\frac{2}{49} & \frac{7}{44} & \frac{7}{11} & \frac{1}{6} \\
\frac{1}{98} & \frac{1}{25} & \frac{5}{17} & \frac{2}{3}
\end{bmatrix}
= \begin{bmatrix}
\frac{4}{3} & \frac{1}{4} & \frac{1}{16} & \frac{1}{48} \\
\frac{1}{3} & 1 & \frac{1}{4} & \frac{1}{12} \\
\frac{1}{12} & \frac{1}{4} & 1 & \frac{1}{3} \\
\frac{1}{48} & \frac{1}{16} & \frac{1}{4} & \frac{4}{3}
\end{bmatrix}
\times \begin{bmatrix}
\frac{9}{\Pi} & \frac{2}{\Pi} & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{2}{\Pi} & \frac{9}{\Pi}
\end{bmatrix}
\]

(a) The Optimal Mechanism   \hspace{1cm} (b) \ G_{3,\frac{1}{4}}   \hspace{1cm} (c) Consumer Interaction

Table 5.1: The optimal mechanism for consumer \( c \) characterized by loss-function \( l(i, r) = |i - r| \) and side-information \( S = \{0, 1, 2, 3\} \). \( n = 3, \alpha = 1/4 \).

There are two reasons why this is undesirable. First, the naive solution results in the release of several re-randomizations of the query result—this allows colluding consumers to combine their results and cancel out the noise leading to a degradation in privacy; see [McSherry and Talwar, 2007] for a discussion.

Second, solving the linear program that identifies the optimal mechanism for a user requires the knowledge of the consumer’s parameters; knowledge that is often unavailable when the decision of which mechanism to deploy is made. Consider a report published on the Internet. It is not clear who the information consumers are going to be.

Our main result works around these issues successfully.

**Theorem 5.4.** Consider a database \( d \), count query \( q \), \( k \) consumers and privacy levels \( \alpha_1 < \ldots < \alpha_k \). There exists a mechanism \( M \) that constructs \( k \) results \( r_1 \ldots r_k \), and releases result \( r_i \) to the \( i \)th information consumer, such that:

1. (Collusion-Resistance) Mechanism \( M \) is \( \alpha_{i'} \)-differentially private for any set \( I \) of colluding information consumers who combine their results. Here, \( C \subseteq \{1 \ldots k\} \) and \( i' = \min\{j : j \in C\} \).

2. (Simultaneous Utility Maximization) Suppose that the \( i \)th consumer is rational and
interacts optimally with the mechanism (as described in Section 5.2.4), then its utility is equal to that of the differentially private mechanism tailored specifically for it (the mechanism from Section 5.2.5).

We now describe the release mechanism $M$. The $i$th stage of the mechanism $M_i$ is just the $\alpha_i$-geometric mechanism. We shall prove in Lemma 5.10, that for any $\alpha > \beta$, the $\alpha$-geometric mechanism can be derived from the $\beta$-geometric mechanism: that is there is an implementable mechanism $T_{\alpha,\beta}$ such that if we use $T_{\alpha,\beta}$ to reinterpret results given by the $\beta$-geometric mechanism, we get the $\alpha$-geometric mechanism. The query results $r_i$ are not generated independently of each other, they are obtained by successive perturbations: the result $r_i$ of mechanism $M_i$ is given as input to the mechanism $T_i = T_{\alpha_i,\alpha_{i+1}}$. Hence, the $(i + 1)$th stage mechanism $M_{i+1}$ is just the $\alpha_{i+1}$-geometric mechanism. This specifies how the noise added to the query results is correlated. We describe the mechanism formally in Algorithm 3. In Section 5.4.1 we show that it is collusion-resistant.

Consumer $i$ interacts optimally with the published query result $r_i$ to get a result tailored specifically for it. In Section 5.4.2, we prove that the interaction yields optimal utility for the consumer. The main idea is that the optimal mechanism can be factored into two parts – The first is a database specific mechanism which has access to the database but not to the user parameters. In our case this is the $\alpha_i$-geometric mechanism. The second is the user specific mechanism, which has access to the user loss-function and side-information and the perturbed query result (given by the first mechanism), but not to the database itself. Table 5.1 shows these two factors of the optimal mechanism discussed in Section 5.2.5.

We briefly discuss proof techniques: Section 5.3 completely characterizes mechanisms derivable from the geometric mechanism using linear algebraic techniques. Section 5.4 applies this characterization twice: the first application shows that a $\alpha$-geometric mechanism can be derived by re-randomizing the output of a $\beta$-geometric
mechanism so long as \( \alpha > \beta \). The second application shows that the mechanism induced by the interaction of a rational information consumer with the geometric mechanism is an optimal solution to the linear program mentioned in Section 5.2.5.

5.2.7 Comparison with Bayesian Information Consumers

An alternative to the Minimax decision rule is the Bayesian decision rule. Ghosh, Roughgarden, and Sundararajan [2009] prove an analogous result to Theorem 5.4 for all Bayesian information consumers. We briefly compare the models and the proof techniques.

The main distinction between the two models is their treatment of side-information. The Bayesian model requires agents to have a prior over all possible scenarios (true query results). Often, in practice, agents do not behave consistent with the preferences of the Bayesian model, perhaps because they find it hard to come up with meaningful priors [Mas-Colell, Whinston, and Green, 1995, Example 6.B.2, 6.B.3], or are genuinely risk-averse [Mas-Colell, Whinston, and Green, 1995, Section 6.3].

As discussed by Ghosh, Roughgarden, and Sundararajan [2009], Bayesian information consumers employ deterministic post-processing, unlike minimax information consumers which employ randomized post-processing (For example, see Table 5.1). Handling this extra complexity requires us to construct a broader characterization of mechanisms derivable from the geometric mechanism—Section 5.3 presents a complete characterization in terms of a simple condition on the probability masses \( x_{i-1,j}, x_{i,j}, x_{i+1,j} \). Our proof avoids the LP based techniques and counting arguments of Ghosh, Roughgarden, and Sundararajan [2009], and consequentially strictly generalizes and gives a simpler proof of the main result of that chapter. In addition, our characterization enables us to release data at multiple levels of privacy in a collusion-resistant manner.
5.2.8 Related Work

A recent thorough survey of the state of the field of differential privacy is given by Dwork [2008]. Dinur and Nissim [2003], Dwork, McSherry, and Talwar [2007] establish upper-bounds on the number of queries that can be answered with reasonable accuracy. Most of the differential privacy literature circumvents these impossibility results by focusing on interactive models where a mechanism supplies answers to only a sub-linear (in $n$) number of queries. Count queries (e.g. [Dinur and Nissim, 2003, Dwork and Nissim, 2004]) and more general queries (e.g. [Dwork, McSherry, Nissim, and Smith, 2006, Nissim, Raskhodnikova, and Smith, 2007]) have been studied from this perspective.

Hardt and Talwar [2010] give tight upper and lower bounds on the amount of noise needed to ensure differential privacy for $d$ non-adaptive linear queries, where the database is a vector in $\mathbb{R}^n$. Hay, Rastogi, Miklau, and Suciu [2010] give a way to increase accuracy of answering multiple related queries while ensuring that the query results follow consistency constraints.

Blum, Ligett, and Roth [2008] focus attention to count queries that lie in a restricted class; they obtain non-interactive mechanisms that provide simultaneous good accuracy (in terms of worst-case error) for all count queries from a class with polynomial VC dimension. Kasiviswanathan, Lee, Nissim, Raskhodnikova, and Smith [2008] give further results for privately learning hypotheses from a given class.

The use of abstract utility functions in McSherry and Talwar [2007] has a similar flavor to our use of loss-functions, though the motivations and goals of their work and ours are unrelated. Motivated by pricing problems, McSherry and Talwar [2007] design differentially private mechanisms for queries that can have very different values on neighboring databases (unlike count queries); they do not consider users with side information and do not formulate a notion of mechanism optimality (simultaneous or otherwise).

Our formulation of the multiple privacy levels is similar to Xiao, Tao, and Chen
However, they use random output perturbations for preserving privacy, and do not give formal guarantees about differential privacy.

5.3 Characterizing Mechanisms Derivable from the Geometric Mechanism

In this section we give a characterization of all mechanisms that can be derived from the geometric mechanism. Recall that differential privacy imposes conditions on every two consecutive entries \((x_1, x_2)\) of every column: \(x_1 \geq \alpha x_2\) (and \(x_2 \geq \alpha x_1\)). Our characterization imposes syntactically similar conditions on every three consecutive entries \((x_1, x_2, x_3)\) in a column: \((x_2 - \alpha \cdot x_3) \geq \alpha(x_1 - \alpha \cdot x_2)\). Neither condition implies the other. This characterization is both necessary and sufficient for any differentially private mechanism to be derivable from the geometric mechanism.

We defined feasible consumer interactions in Section 5.2.4. A slightly different way of representing these is to arrange the probability masses in a \(n \times n\) matrix \((T_{r,r'})_{r,r' \in \mathbb{N}}\). We say that a matrix is (row) stochastic if the sum of elements in each row is 1 and all elements are non-negative. We say that a matrix is a generalized (row) stochastic matrix if the sum of elements in each row is 1, but with no condition on individual entries. If the deployed mechanism is given by the matrix \(y\), and the reinterpretation by the matrix \(T\), then the new mechanism is given by the matrix \(x = y \cdot T\).

We define a version of the Geometric Mechanism whose range is restricted to \(\{0, \ldots, n\}\), which will be easier to work with since it can be easily represented as a matrix.

**Definition 5.5** (Range-Restricted Geometric Mechanism). For a given privacy parameter \(\alpha\), when the true query result is \(k \in [0, n]\), the mechanism outputs \(Z(k)\) where
$Z(k)$ is a random variable with the following distribution for each integer $z$:

$$Pr[Z(k) = z] = \begin{cases} 
\frac{1}{1 + \alpha} \cdot \alpha^{|z-k|} & \text{if } z \in \{0, n\} \\
\frac{1 - \alpha}{1 + \alpha} \cdot \alpha^{|z-k|} & \text{if } 0 < z < n \\
0 & \text{otherwise.}
\end{cases}$$

This mechanism is equivalent to the geometric mechanism in the sense that we can derive this from the geometric mechanism and derive the geometric mechanism from its range-restricted version. We shall refer to both as the Geometric Mechanism and denote the matrix by $G_{n,\alpha}$. (Table 5.2).

For ease of notation, we shall denote by $G'_{n,\alpha}$ the matrix obtained by multiplying the columns 1 and $n$ of $G_{n,\alpha}$ by $(1 + \alpha)$ and all other entries by $\frac{1 + \alpha}{1 - \alpha}$. Table 5.2 shows the matrices of $G_{n,\alpha}$ and $G'_{n,\alpha}$. We are now ready to state the characterization.

\[
G_{n,\alpha} = \begin{bmatrix}
\frac{1}{1 - \alpha} & 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\
\frac{1}{1 - \alpha} & \alpha & 1 & \alpha & \ldots & \alpha^{n-2} \\
\frac{1}{1 - \alpha} & \alpha^2 & \alpha & 1 & \ldots & \alpha^{n-2} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\frac{1}{1 - \alpha} & \alpha^{n-1} & \cdots & 1 & \frac{1}{1 - \alpha} \\
\end{bmatrix}
\]

\[
G'_{n,\alpha} = \begin{bmatrix}
1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\
\alpha & 1 & \alpha & \ldots & \alpha^{n-2} \\
\alpha^2 & \alpha & 1 & \ldots & \alpha^{n-2} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\alpha^{n-1} & \alpha^{n-2} & \cdots & 1 & \frac{1}{1 - \alpha} \\
\end{bmatrix}
\]

Table 5.2: The Range Restricted Geometric Mechanism

**Theorem 5.6.** Suppose $M$ is any oblivious differentially private mechanism. Then $M$ can be derived from the geometric mechanism if and only if every three consecutive entries $x_1, x_2, x_3$ in any column of $M$ satisfy $(x_2 - \alpha x_1) \geq \alpha(x_3 - \alpha x_2)$.

The key insight is to think of each column in $M$ and in $G_{n,\alpha}$ as a vector. Looking at the problem through this linear algebraic lens, we see that deriving $M$ from $G_{n,\alpha}$ amounts to proving that each column of $M$ lies in the convex hull of the (vectors which
form the) columns of $G_{n,\alpha}$. In Lemma 5.7, we show that $G_{n,\alpha}$ is non-singular, hence each column of $M$ can be represented as a linear combination of columns of $G_{n,\alpha}$.

**Lemma 5.7.** $\det(G_{n,\alpha}) > 0$.

*Proof.* Since $G'_{n,\alpha}$ can be obtained by multiplying each entry in the first and last column of $G_{n,\alpha}$ by $(1 + \alpha)$ and entries in all other columns by $\frac{1 + \alpha}{1 - \alpha}$, $\det G'_{n,\alpha} = (1 + \alpha)^2 \left(\frac{1 + \alpha}{1 - \alpha}\right)^{n-2} \det G_{n,\alpha}$. Hence, we only need to prove that $\det G'_{n,\alpha} > 0$. We prove this by induction on $n$. For $n = 2$, an explicit calculation yields $G'_{2,\alpha} = (1 - \alpha^2)$. For the general case, perform the column transformation $C_1 \leftarrow C_1 - \alpha C_2$ on $G'_{n,\alpha}$. Expanding on the first column gives us $\det G'_{n,\alpha} = (1 - \alpha^2) \det G'_{n-1,\alpha}$. Hence, by induction, $\det G'_{n,\alpha} = (1 - \alpha^2)^{n-1}$. \hfill $\square$

We need to show that each column of $M$ is actually a convex combination of columns of $G$. We can write $M = G_{n,\alpha} \cdot T$ for some matrix $T$. Hence, $T = G_{n,\alpha}^{-1} \cdot M$. Note that $G_{n,\alpha}$ and $M$ are both generalized stochastic matrices. Since the set of all non-singular generalized stochastic matrices forms a group [Poole, 1995], $G_{n,\alpha}^{-1}$ is a generalized stochastic matrix. And since generalized stochastic matrices are closed under multiplication, $T$ is also a generalized stochastic matrix and is uniquely defined. All we need to prove is that all entries in $T$ are non-negative. We shall use Cramer’s Rule to calculate the entries of $T$ and complete the proof.

Given a $n \times n$ matrix $G$ and a vector $x = (x_1, \ldots, x_n)^t$, define $G(i, x)$ as the matrix where the $i^{th}$ column of $G$ has been replaced by $x$.

Let $t_j$ be the $j^{th}$ column of $T$. $t_{i,j}$ denotes the $i, j$ entry in $T$. Observe that, $G_{n,\alpha} \cdot t_j = m_j$. By Cramer’s Rule, we get that $t_{i,j} = \frac{\det G_{n,\alpha}(i, m_j)}{\det G_{n,\alpha}}$. To calculate this, we shall explicitly calculate the value of $\det G_{n,\alpha}(i, m_j)$.

**Lemma 5.8.** Given $G_{n,\alpha}$ and a vector $x = (x_1, \ldots, x_n)^t$:

1. $\det G_{n,\alpha}(1, x) > 0$ iff $x_1 > \alpha x_2$
2. $\det G_{n,\alpha}(n, x) > 0$ iff $x_n > \alpha x_{n-1}$

3. For $2 \leq i \leq n - 1$ : $\det G_{n,\alpha}(i, x) \geq 0$ if and only if $(x_2 - \alpha x_1) \geq \alpha(x_3 - \alpha x_2)$

Since $M$ is a differentially-private mechanism, every column of $M$ satisfies $x_1 > \alpha x_2$ and $x_n > \alpha x_{n-1}$. Hence, when $M$ satisfies the condition that for every three consecutive entries $x_1, x_2, x_3$ in any column $(x_2 - \alpha x_1) \geq \alpha(x_3 - \alpha x_2)$, then $s_{i,j} \geq 0$ for all $i, j$. This proves that $M$ can be derived from the geometric mechanism.

To prove the converse, suppose that there is a column $c$ and row $i$ of $M$ such that $((1 + \alpha^2)m_{i,j} - \alpha(m_{i-1,j} + m_{i+1,j})) < 0$, then $s_{i,c} = \det G(i, m_c)/\det G < 0$. This says that $M$ cannot be derived from $G$. This completes the proof of Theorem 5.6.

We now prove Lemma 5.8, using similar column transformations as we used in Lemma 5.7 to calculate $\det G_{n,\alpha}(i, x)$ for an arbitrary vector $x$.

**Lemma 5.9.** Given $G_{n,\alpha}$ and a vector $x = (x_1, \ldots, x_n)^t$:

1. $\det G_{n,\alpha}(1, x) > 0$ iff $x_1 > \alpha x_2$

2. $\det G_{n,\alpha}(n, x) > 0$ iff $x_n > \alpha x_{n-1}$

3. For $2 \leq i \leq n - 1$ : $\det G_{n,\alpha}(i, x) \geq 0$ if and only if $(x_2 - \alpha x_1) \geq \alpha(x_3 - \alpha x_2)$

**Proof.** We will prove the above properties for $G'_{n,\alpha}$. Since, $G'_{n,\alpha}$ is obtained from $G_{n,\alpha}$ by multiplying columns with positive reals, the properties above will continue to hold for $G_{n,\alpha}$. We divide the proof into cases depending on the value of $i$:

1. $i = 1$ : For $j$ from $n$ down to 3 do the column transformation $C_j \leftarrow C_j - \alpha C_{j-1}$

   to get that $\det G'_{n,\alpha}(1, x) = (1 - \alpha^2)^{n-2} \begin{vmatrix} x_1 & \alpha \\ x_2 & 1 \end{vmatrix} = (1 - \alpha^2)^{n-2}(x_1 - \alpha x_2)$. Hence,

   $\det G'_{n,\alpha}(1, x) > 0 \iff (x_1 > \alpha x_2)$. 

2. For $i = n$: Similarly, for $j$ from 1 to $n - 3$, do the column transformation $C_j \leftarrow C_j - \alpha C_{j+1}$ to get that $G'_{n,\alpha}(n, x) = (1 - \alpha^2)^{n-2} \begin{vmatrix} 1 & x_{n-1} \ \\ \alpha & x_n \end{vmatrix} = (1 - \alpha^2)^{n-2}(x_n - \alpha x_{n-1})$. Hence, $\det G'_{n,\alpha}(n, x) > 0 \iff (x_n > \alpha x_{n-1})$.

3. For $2 \leq i \leq n - 1$: Similarly, for the general case, $j$ from 1 to $i - 2$, we perform the transformations $C_j \leftarrow C_j - \alpha C_{j+1}$ and for $j$ from $n$ down to $i + 2$ set $C_j \leftarrow C_j - \alpha C_{j-1}$. From this we get that $\det G'_{n,\alpha}(i, x) = (1 - \alpha^2)^{n-3} \begin{vmatrix} 1 & x_{i-1} & \alpha^2 \\ \alpha & x_i & \alpha \\ \alpha^2 & x_{i+1} & 1 \end{vmatrix} = (1 - \alpha^2)^{n-2}(1 + \alpha^2)x_i - \alpha(x_{i-1} + x_{i+1})$. Hence, $\det G'_{n,\alpha}(i, x) > 0 \iff (x_2 - \alpha x_1) \geq \alpha(x_3 - \alpha x_2)$.

\[\square\]

5.4 Applications of the Characterization

We show two applications of the characterization result of Theorem 5.6. The first one gives us a way to simultaneously release data to consumers at different levels of privacy. As a second application we show how to obtain a optimal mechanism for an information consumer without knowing its parameters.

5.4.1 Information-Consumers at Different Privacy Levels

Suppose we want to release the answer of the query to different information consumers. We represent the level of privacy of a consumer $c$ by the privacy parameter $\alpha_c$. Given true result $r$, we will release $r_c$ to consumer $c$ such that the mechanism is $\alpha_c$-differentially private. We expect that consumers at different levels of privacy do not share query results with each other which is enforced via, say, non-disclosure agreements. Even when they do share data, we want our mechanism to be collusion-resistant and not leak privacy—the colluding group should not get any more information about the database.
than the consumer with access to the least private result i.e., the one with the smallest $\alpha$.

We now describe a mechanism that achieves this. The next lemma gives us a way to “add” more privacy to an existing geometric mechanism.

**Lemma 5.10.** For two privacy parameters $\alpha \leq \beta$, the geometric mechanism $G_{n,\beta}$ can be derived from the mechanism $G_{n,\alpha}$ i.e., there exists a stochastic matrix $T_{\alpha,\beta}$ such that $G_{n,\beta} = G_{n,\alpha} \cdot T_{\alpha,\beta}$.

**Proof.** Theorem 5.6 states that $G_{n,\beta}$ can be derived from $G_{n,\alpha}$ if and only if for every three consecutive entries $x_1, x_2, x_3$ in any column of $G_{n,\beta}$, $(x_2 - \alpha x_1) \geq \alpha(x_3 - \alpha x_2)$. We check this condition for each of the three forms that consecutive entries in each row of $G_{\beta,n}$ can have:

1. $(\beta^i, \beta^{i+1}, \beta^{i+2}) : (1 + \alpha^2)\beta^{i+1} - \alpha(\beta + \beta^{i+2}) = \beta^i(\beta + \alpha^2 \beta - \alpha - \alpha \beta^2) = \beta^i(\beta - \alpha)(1 - \alpha \beta) > 0$.

2. $(\beta, 1, \beta) : (1 + \alpha^2)1 - \alpha(\beta + \beta) = 1 + \alpha^2 - 2\alpha \beta > (1 - \alpha)^2 > 0$.

3. $(\beta^{i+2}, \beta^{i+1}, \beta^i) : (1 + \alpha^2)\beta^{i+1} - \alpha(\beta + \beta^{i+2}) = \beta^i(\beta + \alpha^2 \beta - \alpha - \alpha \beta^2) = \beta^i(\beta - \alpha)(1 - \alpha \beta) > 0$.

This shows that $T_{\alpha,\beta} = G_{n,\alpha}^{-1} \cdot G_{n,\beta}$ is a stochastic matrix. \qed

The release mechanism is given in Algorithm 3. We conclude the section by proving that Algorithm 3 is collusion-resistant.

**Lemma 5.11.** Any subset $C = \{c_1 < \cdots < c_t\} \subseteq \{1, \ldots, k\}$ of colluding information consumers who have access to query results $R(C) = \{r_{c_1} \cdots r_{c_k}\}$, released at privacy levels $\alpha_{c_1}, \ldots, \alpha_{c_k}$, respectively, can only reconstruct as much information about the database $d$ by combining their results as $c_1$ can working alone.
Algorithm 3: Releasing Query Result to Consumers at Multiple Levels of Trust.

**Input:** True Query Result \( r \), \( k \) privacy levels given by parameters \( \alpha_1 < \alpha_2 < \ldots < \alpha_k \).

**Output:** Query Results \( r_1, r_2, \ldots, r_k \) to be released.

Define \( T_1 = G_{\alpha_1,n} \).

**for** \( 1 \leq i \leq k \) **do**

Compute post-processing matrix \( T_{i+1} \) such that \( G_{\alpha_{i+1},n} = G_{\alpha_i,n} \cdot T_{i+1} \).

**end**

By Lemma 5.10, each \( T_i \) is a stochastic matrix. Hence, we can think of \( T_i \) as a mechanism – Given any input \( k \) we sample from the probability distribution given by the \( k^{th} \) row of \( T_i \) which we represent by \( T_i(k) \).

Let \( r_0 = r \).

**for** \( 1 \leq i \leq k \) **do**

\( r_i = T_i(r_{i-1}) \) is obtained by treating \( r_{i-1} \) as the true query output and applying mechanism \( T_i \) to it.

**end**

Release the query results \( r_1, r_2, \ldots, r_k \) to consumers at privacy levels \( \alpha_1, \ldots, \alpha_k \).

**Proof.** The matrix \( G_{n,\alpha_1} \) and post-processing matrices \( T_{\alpha_{c_i},\alpha_{c_{i+1}}} \) can be calculated by anyone. Hence, given the random coin tosses made by the algorithm, Lemma 5.10 shows that \( r_{c_j} \) can be obtained from \( r_{c_i} \) for \( c_j > c_i \). Given \( r_{c_1} \), having access to \( R(C) \) can at most reveal information about these coin tosses that Algorithm 3 made. Since, these coin tosses do not depend on the database, any information about the database that is reconstructed from \( R(C) \) can also be reconstructed by consumer \( c_1 \) (who has access to result \( r_{c_1} \)) alone.

\( \square \)

5.4.2 Universal Utility Maximizing Mechanisms

We now prove that if we deploy the geometric mechanism (Definition 5.5), then the interaction of every information consumer will yield a mechanism that is optimal for
that consumer. Since, the geometric mechanism is not dependent on any information consumer’s loss-function or side information, it is simultaneously optimal for all of them.

Our result proves that all optimal mechanisms can be derived from the geometric mechanism. However, there do exist differentially private mechanisms (which are not optimal for any information consumer) that cannot be derived from the geometric mechanism. We give an example of such a mechanism in Appendix 5.6.

The first part of the proof shows that every two adjacent rows of every optimal mechanism must satisfy certain condition; if it does not, we can perturb the mechanism in a way to yield a differentially private mechanism with strictly better utility. The second part of the proof leverages this lemma and the characterization from Theorem 5.6 to complete the proof of Theorem 5.4.

**Lemma 5.12.** For every monotone loss-function \( L(l, S) = \max_{i \in S} \sum_{r \in N} l(i, r) \cdot x_{i,r} \), there exists an optimal mechanism \( x \) such that for every two adjacent rows \( i, i + 1 \) of this mechanism, there exist column indices \( c_1 \) and \( c_2 \) such that:

1. \( \forall j \in 1...c_1 : \alpha x_{i,j} = x_{i+1,j} \)
2. \( \forall j \in c_2...n : x_{i,j} = \alpha x_{i+1,j} \)
3. Either \( c_2 = c_1 + 1 \) or \( c_2 = c_1 + 2 \).

**Proof.** We define the function \( L' : M \to \mathbb{R} \) given by \( L'(x) = \sum_{i \in N} \sum_{r \in N} x_{i,r} |i-r| \). Consider the total order \( \succ \) on \( \mathbb{R}^2 \) given by \( (a, b) \succ (c, d) \iff \{(a > c) \text{ or } (a = c \text{ and } b > d)\} \).

Let \( x \) be an optimal mechanism for the loss-function \((L, L')\) according to the order defined above. The idea here is that there are multiple mechanisms that optimize \( L \) and using \( L' \) we isolate the ones with the property that we want. We prove by contradiction that \( x \) satisfies the constraints given above.

Assume otherwise. Then there exist rows \( i, i+1 \) and columns \( j, k ; k > j \) such that \( \alpha x_{i,j} < x_{i+1,j} \) and \( \alpha x_{i+1,k} < x_{i,k} \). We shall construct a differentially private mechanism
y for which \((L_y, L'_y)\) is strictly smaller than \((L_x, L'_x)\) which is a contradiction since we assumed that \(x\) minimized \((L, L')\).

We divide the proof into two cases: \(i \leq (j+k)/2\) and \(i > (j+k)/2\). Consider the case \(i \leq (j+k)/2\) first. For \(i' \in \{1 \ldots i\}\) set \(y_{i',j} \leftarrow x_{i',j} + \delta x_{i',k}\) and \(y_{i',k} \leftarrow (1-\delta)x_{i',k}\). For all other values set \(y_{l,m} = x_{l,m}\). We first show that \(y\) is a differentially private mechanism. Let the set of changed elements \(C = \{y_{l,m} : m \in \{j,k\} \text{ and } l \leq i\}\). The set of unchanged elements \(U\) is all the remaining \(y_{l,m}\). All privacy constraints involving elements only from \(U\) are satisfied since they were satisfied in \(M\). The privacy constraints involving only elements in \(C\) continue to hold since they are the same linear combinations of corresponding elements from \(M\). We only need to check that the privacy constraints are satisfied when one element is from \(C\) and another from \(U\). But this only happens for \(y_{i,j}, y_{i+1,j}\) and \(y_{i,k}, y_{i+1,k}\). By assumption, \(\alpha x_{i,j} < x_{i+1,j}\) and \(\alpha x_{i+1,k} < x_{i,k}\). We can choose a small enough \(\delta\) such that \(\alpha y_{i,j} = \alpha(x_{i,j} + \delta x_{i,k}) < x_{i+1,j} = y_{i+1,j}\) and \(\alpha y_{i,k} = \alpha(1-\delta)x_{i,k} < x_{i+1,k} = y_{i+1,k}\). Also, for \(m \in \{j,k\}\), \(y_{i,m} > x_{i,m} > \alpha x_{i+1,m} = y_{i+1,m}\). This proves that \(y\) satisfies differential privacy.

Now, we shall prove that \(y\) has strictly smaller loss than \(x\). For any row \(r \in \{1, \ldots, i\}\), the change in loss due to row \(r\) is

\[
\sum_{i \in N} l(r,i)x_{r,i} - \sum_{i \in N} l(r,i)y_{r,i}
= (l(r,j)x_{r,j} + l(r,k)x_{r,k}) - (l(r,j)y_{r,j} + l(r,k)y_{r,k})
= (l(r,j)x_{r,j} + l(r,k)x_{r,k})
- (l(r,j)(x_{r,j} + \delta x_{r,k}) + l(r,k)(x_{r,k} - \delta x_{r,k})
= \delta x_{r,k}(l(r,k) - l(r,j))
\geq 0 \quad \text{since } l(i,j) \text{ is monotonic in } |i-j|.
\]

The total loss \(L = \max_{r \in S} \sum_{i \in N} l(i,r) \cdot x_{i,r}\) and since \(L_x \geq L_y\). Also \(\sum_{i} \sum_{r} x_{i,r} \cdot |i-r| > \sum_{i} \sum_{r} y_{i,r} \cdot |i-r|\). This means that \((L_x, L'_x) > (L_y, L'_y)\). But \(x\) was an optimal
mechanism with respect to \( \succ \). This gives us a contradiction.

The proof for the case \( i > (j + k)/2 \) is similar. For \( i \geq i' \) set \( y_{i',k} \leftarrow x_{i',k} + \delta x_{i',j} \) and \( y_{i',j} \leftarrow (1 - \delta)x_{i',j} \). The same arguments as above now hold for this definition of \( y \) as well. \( \square \)

We are now ready to prove Theorem 5.4. We state it again for convenience.

**Theorem 5.4.** Consider a database \( d \), count query \( q \), \( k \) consumers and privacy levels \( \alpha_1 < \ldots < \alpha_k \). There exists a mechanism \( M \) that constructs \( k \) results \( r_1 \ldots r_k \), and releases result \( r_i \) to the \( i \)th information consumer, such that:

1. **(Collusion-Resistance)** Mechanism \( M \) is \( \alpha_{i'} \)-differentially private for any set \( I \) of colluding information consumers who combine their results. Here, \( C \subseteq \{1 \ldots k\} \) and \( i' = \min\{j : j \in C\} \).

2. **(Simultaneous Utility Maximization)** Suppose that the \( i \)th consumer is rational and interacts optimally with the mechanism (as described in Section 5.2.4), then its utility is equal to that of the differentially private mechanism tailored specifically for it (the mechanism from Section 5.2.5).

**Proof.** Algorithm 3 is used to deploy geometric mechanism at different levels of privacy. Lemma 5.10 shows that it is always possible to deploy geometric mechanism this way. This proves that the deployed mechanisms are differentially private. Lemma 5.11 proves that the release is \( \alpha_{c_i'} \)-differentially private even for any set \( C \) of colluding consumers, where \( i' = \min\{j : j \in C\} \). This completes the proof of part 1.

To prove part 2, we concentrate on a single trust level with privacy parameter \( \alpha \). We prove the result by contradiction. Assume there is an information consumer \( c \) with loss-function \( l \) and side-information \( S \), whose interaction with \( G_{n,\alpha} \) does not optimize its loss. Let \( M \) be an optimal differentially private mechanism for \( c \) that satisfies Lemma 5.12. Since, \( c \) cannot optimize its loss by interacting with \( G_{n,\alpha} \), \( M \) cannot be
derived from the geometric mechanism. We prove that this implies that \( M \) is infeasible which is a contradiction.

We know from Theorem 5.6 that there exists a column \( j \) of \( M \) and rows \( i, i+1, i+2 \), such that the three entries \( x_{i,j}, x_{i+1,j}, x_{i+2,j} \) satisfy

\[
(1 + \alpha^2)x_{i+1,j} - \alpha(x_{i,j} + x_{i+2,j}) < 0. \tag{5.2}
\]

Recall the pattern of every pair of adjacent rows of \( M \) from Lemma 5.12. Let \( k \) be the unique column that satisfies \( \alpha x_{i,k} < x_{i+1,k} \) and \( \alpha x_{i+1,k} < x_{i,k} \), or if there is no such column, let it be the last column such that \( \alpha x_{i,j} = x_{i+1,j} \). Let \( a = \sum_{l<k} x_{i,l} \), \( b = x_{i,k} \), \( b' = x_{i+1,k} \), \( b'' = x_{i+2,k} \) and \( c = \sum_{l>k} x_{i,l} \). Rewrite Equation (5.2) to get:

\[
0 \leq x_{i+1,j} - \alpha x_{i+2,j} < \alpha(x_{i,j} - \alpha x_{i+1,j}) \implies x_{i,j} > \alpha x_{i+1,j} \]

Thus, by Lemma 5.12, \( k \geq j \). We now claim that:

\[
(1 + \alpha^2)b' - \alpha(b + b'') < 0 \tag{5.3}
\]

This is true from Equation (5.2) if \( k = j \). Otherwise rewrite Equation (5.2) to get:

\[
0 \leq x_{i+1,j} - \alpha x_{i,j} < \alpha(x_{i+2,j} - \alpha x_{i+1,j}) \implies x_{i+2,j} > \alpha x_{i+1,j}. \]

Thus, by Lemma 5.12, it must be that \( \alpha \cdot b'' = b \). Further, by privacy \( b \geq \alpha b' \) and so, \( b > \alpha^2 b' \). This proves the claim.

Because \( M \) is a generalized stochastic matrix, \( \sum_{l} x_{i,l} = \sum_{l} x_{i+1,l} = 1 \). Thus, \( a + b + c = 1 \) and \( \alpha \cdot a + b' + c/\alpha = 1 \). Using these equations, we have:

\[
a = \frac{1 - b - \alpha + b'\alpha}{1 - \alpha^2} \quad \text{and} \quad c = \frac{\alpha - \alpha^2 + b\alpha^2 - b'\alpha}{1 - \alpha^2} \tag{5.4}
\]
We now prove that $M$ is not feasible.

$$
\sum_l x_{i+2,l} \geq \alpha^2 \cdot a + b'' + c/\alpha^2
$$

$$
= \frac{\alpha^3 - b\alpha^3 - \alpha^4 + b'\alpha^4 + b''\alpha - b''\alpha^3 + 1 - \alpha - b' + b\alpha}{\alpha(1 - \alpha^2)}
$$

$$
= \frac{1 - \alpha + \alpha^2}{\alpha} + \frac{(b + b'\alpha - b'(1 + \alpha^2)}{\alpha}
$$

$$
> 1
$$

The first step is from Equation (5.2) and Lemma 5.12, the second is by Equation (5.4), the third is by rearranging and the fourth holds because the first summand is always at least 1 and the second is strictly positive by Equation (5.3).

5.5 There always exists an Optimal Mechanism that is Oblivious

In Section 5.2 we restricted attention to oblivious mechanisms. While natural mechanisms (such as the Laplace mechanism from [Dwork, McSherry, Nissim, and Smith, 2006]) are usually oblivious, we now justify this restriction from first principles. Specifically, we show that for every information consumer with a loss-function over databases and side information over query results, there exists a oblivious loss-function$^5$ and side-information, such that the optimal utility with the oblivious loss-function is no more than the optimal utility with the loss-function that depends on the database.

Consider a non-oblivious mechanism $x$. For the minimax information consumer with loss-function $l$ over databases and side information $S \subseteq \{0, 1, \ldots, n\}$, the utility of this mechanism is given by

$$
\max_{d \in S \subseteq D^n} \sum_{r \in N} x_{d,r} \cdot l(f(d), r)
$$

The following lemma proves that obliviousness is without loss of generality i.e. there always exists an oblivious mechanism whose loss is lower than or equal to the loss of the best non-oblivious mechanism.

$^5$We say a loss-function is oblivious if it depends only on output given by the mechanism and on the true query output and not on the database
Lemma 5.13. Fix a database size \( n \geq 1 \) and privacy level \( \alpha \). For every minimax information consumer with loss-function \( l \) and side information \( S \subseteq \{0,1,\ldots,n\} \), there is an \( \alpha \)-differentially private mechanism that minimizes the objective function (5.5) and is also oblivious.

Proof. We shall now construct a differentially privacy mechanism \( x' \) that is oblivious and whose loss is not greater than the loss of \( x \). This will prove our assertion.

We construct a partition \( E \) of all the databases, according to the query output. All databases that have the same query output belong to the same subset of the partition. For a database \( d \), let \( E(d) = \{d' : q(d) = q(d')\} \). For \( r \in \mathbb{N} \) and \( d \in D^n \), define \( x'_{E(d),r} = \text{avg}_{d' \in E(d)} x_{d',r} \). It is clear that \( x' \) is an oblivious mechanism.

First we show that \( x' \) is \( \alpha \)-differentially private. Fix two databases \( d_1, d_2 \in D^n \) such that \( d_1 \) and \( d_2 \) differ in exactly one row; We need to show that \( \alpha x'_{E(d_1),r} \leq x'_{E(d_2),r} \).

Assume \( f(d_1) \neq f(d_2) \), otherwise the proof is trivial.

For any database of \( E(d_1) \), we can generate all its neighbors (databases that differ in exactly one row) in \( E(d_2) \) by enumerating all the ways in which we can change the query result by exactly 1. For instance when \( f(d_1) = f(d_2) + 1 \), pick one of the \( n - f(d_1) \) rows that satisfy the predicate in \( d_1 \) and change its value to one of those that violates the predicate. This process is identical for all databases of \( E(d_1) \), and so for all \( d \in E(d_1) \), the number of neighbors of \( d \) that belong to the set \( E(d_2) \) is the same (does not vary with \( d \)). Similarly, for all \( d \in E(d_2) \), the number of neighbors of \( d \) that belong to the set \( E(d_1) \) is the same.

Consider the following set of inequalities that hold because \( x \) is \( \alpha \)-differentially private: \( d \in E(d_1), d' \in E(d_2) \), where \( d_1 \) and \( d_2 \) are neighbors, \( \alpha x_{dr} \leq x_{d'r} \). By the argument in the above paragraph, all the databases in \( E(d_1) \) appear equally frequently, say \( k_{d_1} \) in the left-hand-side of the above inequality and all the databases in \( E(d_2) \) appear equally frequently, say \( k_{d_2} \) times, in the right-hand-side. Summing the inequalities
we get

\[
\sum_{d \in E(d_1)} \alpha \cdot k_{d_1} \cdot x_{d,r} \leq \sum_{d' \in E(d_2)} k_{d_2} \cdot x_{d',r}
\]

\[
\implies k_{d_1} |E(d_1)| \alpha \avg_{d \in E(d_1)} x_{d,r} \leq k_{d_2} |E(d_2)| \avg_{d' \in E(d_2)} x_{d',r}
\]

But \(k_{d_1} |E(d_1)| = k_{d_2} |E(d_2)|\). To see this consider the bipartite graph between \(E(d_1), E(d_2)\) obtained by looking at each database as a vertex, with an edge connecting two databases if the they are neighbors. Then both the quantities count the number of edges in this bipartite graph.

\[
\implies \alpha \avg_{d \in E(d_1)} x_{d,r} \leq \avg_{d' \in E(d_2)} x_{d',r}
\]

\[
\implies \alpha x_{E(d_1),r} \leq x_{E(d_2),r}
\]

This completes the proof of privacy.

Now we show that \(x'\) does not incur more loss than \(x\). The loss for \(x'\) is given by

\[
\max_{d \in S \subseteq D^n} \sum_{r \in N} x'_{E(d),r} \cdot l(f(d), r).
\]

Suppose the worst loss for \(x'\) occurs for the partition \(E(d_1)\).

\[
L(x') = \sum_{r \in N} x'_{E(d_1),r} \cdot l(f(d_1), r)
\]

\[
= \sum_{r \in N} \avg_{d \in E(d_1)} (x_{d,r}) \cdot l(f(d), r)
\]

\[
\leq \max_{d \in E(d_1)} \sum_{r \in N} x_{d,r} \cdot l(f(d), r)
\]

\[
\leq \max_{d \in S \subseteq D^n} \sum_{r \in N} x_{d,r} \cdot l(f(d), r) = L(x).
\]

This completes the proof.
5.6 A Differentially Private Mechanism that is not derivable from the
Geometric Mechanism

Consider the mechanism $M$ given by the following matrix. $M(i, j)$ gives the probability of returning $j$ when the true query result is $i$. We can verify that $M$ is $\frac{1}{2}$-differentially private.

$$
M = \begin{bmatrix}
\frac{1}{9} & \frac{2}{9} & \frac{4}{9} & \frac{2}{9} \\
\frac{2}{9} & \frac{1}{9} & \frac{2}{9} & \frac{4}{9} \\
\frac{4}{9} & \frac{2}{9} & \frac{1}{9} & \frac{2}{9} \\
\frac{13}{18} & \frac{1}{9} & \frac{1}{18} & \frac{1}{9} \\
\end{bmatrix}
$$

We claim that $M$ cannot be derived from the geometric mechanism. We can explicitly calculate $G_{\frac{3}{2}}^{-1} \cdot M$ to see that $M$ is not derivable from the geometric. Instead we shall use the characterization from Theorem 5.6. If we look at elements $M(0, 1), M(1, 1), M(2, 1)$, then

$$(1 + \alpha^2)M(1, 1) - \alpha(M(0, 1) + M(2, 1)) = 1.25 \times \frac{1}{9} - \frac{1}{2} \times \left( \frac{2}{9} + \frac{2}{9} \right) = \frac{-0.75}{9}.$$

This proves that $M$ cannot be derived from $G_{\frac{3}{2}}$.

5.7 Conclusion

In this chapter, we give a minimax model of utility for information consumers that is prescribed by decision theory. We show that for any particular count query, the geometric mechanism is simultaneously optimal for all consumers, assuming that consumers interact rationally with the output of the mechanism. This is particularly useful in publishing aggregate statistics, like the number of flu infections in a given region, to a wide unknown audience, say on the Internet.

Followup work by Brenner and Nissim [2010] shows that it is impossible to obtain a universally optimal mechanism for sum queries with non-binary individual values, histograms, and two (or more) count queries. An open question is to investigate whether we can get around the impossibility result, for example by providing multiple answers, depending on a parametrization of the class of consumers.
Bibliography


Pedro Domingos and Matthew Richardson. Mining knowledge-sharing sites for viral marketing. *Proceedings 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.


David Kempe, Jon Kleinberg, and Eva Tardos. Maximizing the spread of influence through a social network. *Proceedings 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.


Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 67, 2005.


Thorleif Schjelderup-Ebbe. Contributions to the social psychology of the domestic


Xiaokui Xiao, Yufei Tao, and Minghua Chen. Optimal random perturbation at multiple privacy levels. In *35th International Conference on Very Large Data Bases (VLDB),* pages 814–825, 2009.