Analytic search methods in online social networks

by

Christopher E. Marks

B.S., United States Military Academy
S.M., Massachusetts Institute of Technology

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

©2017 Christopher E. Marks, All rights reserved.

The author hereby grants to MIT and The Charles Stark Draper Laboratory, Inc. permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in any part medium now known or hereafter created.
Analytic search methods in online social networks
by
Christopher E. Marks
Submitted to the Sloan School of Management
on May 16, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract
This thesis presents and evaluates methods for searching and analyzing social media data in order
to improve situational awareness. We begin by proposing a method for network vertex search that
looks for the target vertex by sequentially examining the neighbors of a set of “known” vertices.
Using a dynamic programming approach, we show that there is always an optimal “block” search
policy, in which all of the neighbors of a known vertex are examined before moving on to another
vertex. We provide a precise characterization of the optimal policy in two specific cases: (1) when
the connections between the known vertices and the target vertex are independent, and (2) when the
target vertex is connected to at most one known vertex. We then apply this result to the problem
of finding new accounts belonging to Twitter users whose previous accounts had been suspended
for extremist activity, quantifying the performance of our optimal search policy in this application
against other policies. In this application we use thousands of Twitter accounts related to the
Islamic State in Iraq and Syria (ISIS) to develop a behavioral models for these extremist users.
These models are used to identify new extremist accounts, identify pairs of accounts belonging to
the same user, and predict to whom a user will connect when opening an account. We use this final
model to inform our network search application. Finally, we develop a more general application of
network search and classification that obtains a set of social media users from a specified location
or group. We propose an expand—classify methodology which recursively collects users that have
social network connections to users inside the target location, and then classifies all of the users
by maximizing the probability over a factor graph model. This factor graph model accounts for
the implications of both observed user profile features and social network connections in inferring
location. Using geo-located data to evaluate our method, we find that our classification method
typically outperforms Twitter’s native search methods in building a dataset of Twitter users in a
specific location.

Thesis Supervisor: John Irvine
Title: Draper, Inc.

Thesis Supervisor: Tauhid Zaman
Title: KDD Professor
Acknowledgments

Dr. Jytte Klausen at Brandeis provided access to her ISIS dataset and data collection guidance, from which we built our database of ISIS Twitter accounts. This database directly supported our analyses of online extremist activities, behavioral and predictive models, motivated the development of our network search methodology.

Dr. Mark Abramson, Dr. John Irvine, and Dr. Steve Kolitz from Draper provided valuable direction, contributions, and questions throughout this research. Dr. Kolitz’s guidance and support during this time has been invaluable, and I look forward to future collaborative efforts and continued research.

Lieutenant Krishnan Rajagopalan served as a fellow researcher and friend during our two-year overlap. I appreciate his input and perspectives on my earlier research efforts, and wish him the best in his career in the U.S. Navy.

Many of my colleagues and supervisors in the U.S. Army also provided invaluable support. I especially appreciate the research discussions with Lieutenant Colonel Chris Smith, Lieutenant Colonel Mike Teter, Lieutenant Colonel Matt Benigni, and Lieutenant Colonel Kevin Cummiskey, who all provided feedback and additional insights on my research efforts.

All of the faculty and staff that I have worked with at MIT have been superb. I especially thank my advisor, Professor Tauhid Zaman, for his guidance, mentoring, and support. While it has been a lot of work, Tauhid kept our research interesting, relevant, and fun. I hope we find opportunities for continued collaboration in the future.

Finally, this I could not have devoted the time and focus needed in this program without the loving support and seemingly endless patience of my wife, Carol, and my children, Sarah, MJ, and Bobby. I am eternally grateful, and I look forward to whatever adventures the future has in store.
# Contents

## 1 Introduction

1.1 Problem ................................................................. 17  
1.1.1 Technical Approach ................................................. 18  
1.1.2 Experimentation ...................................................... 18  
1.2 Social Media ............................................................ 18  
1.2.1 Twitter ................................................................. 20  
1.3 Previous Work .......................................................... 25  
1.3.1 Social Networks ....................................................... 26  
1.3.2 Social Media ........................................................... 29  
1.4 Situational Awareness and Intelligence ............................... 31  
1.5 Thesis Organization .................................................... 32

## 2 Optimal User Search in Online Social Networks  
2.1 Background and Motivation .............................................. 35  
2.1.1 Application ............................................................ 37  
2.1.2 Previous Work ........................................................ 37  
2.2 Multi-urn Search Model .................................................. 41  
2.2.1 Dynamic Programming Framework .................................. 44  
2.2.2 Costs ................................................................. 47  
2.3 Key Results .............................................................. 49  
2.3.1 Block Policy Optimality .............................................. 49  
2.3.2 Independent Urns ..................................................... 50  
2.3.3 One Marble .......................................................... 53  
2.3.4 Monotonicity Properties ............................................ 56
3 Finding Online Extremists in Social Networks

3.1 Background and Motivation ........................................... 83
  3.1.1 Overview of Contributions ...................................... 84
  3.1.2 Previous Work .................................................. 86

3.2 Data .............................................................................. 87

3.3 Predicting Account Suspensions ..................................... 88
  3.3.1 Results ............................................................... 89

3.4 Detecting Multiple Accounts .......................................... 91
  3.4.1 Suspended User Behavior ........................................ 92
  3.4.2 Profile Comparison Metrics ................................. 94
  3.4.3 Data Set Construction ........................................... 96
  3.4.4 Data Labeling .................................................... 97
  3.4.5 Classification Model ............................................. 99
  3.4.6 Classification Threshold Sensitivity ...................... 102

3.5 Re-following Model .................................................... 105
  3.5.1 Data ............................................................... 106
  3.5.2 Features .......................................................... 107
  3.5.3 Kernel Logistic Regression .................................. 108
  3.5.4 Performance ....................................................... 109
3.6 Suspended User Search .......................................................... 110
  3.6.1 Suspended User Search Model .......................................... 111
  3.6.2 Initialization ................................................................. 113
  3.6.3 The Discrete Stochastic Search Process ............................. 113
  3.6.4 Search Process Dynamics ................................................. 115
  3.6.5 Analysis: $\bar{\rho} = 0$ .................................................. 119
  3.6.6 Results ................................................................. 123
  3.6.7 Discussion: $\bar{\rho} > 0$ ................................................ 127
3.7 Summary of ISIS Search Contributions ................................. 127

4 Building a Location-Based Set of Social Media Users .................. 129
  4.1 Background and Motivation ................................................ 129
    4.1.1 Previous Work .......................................................... 132
    4.1.2 Expand—Classify Methodology ...................................... 134
  4.2 Classification Model ........................................................ 135
    4.2.1 Factor Graph Representation ........................................ 135
    4.2.2 Model Characteristics ................................................ 137
    4.2.3 Classification Optimization ........................................ 139
  4.3 Choosing the Energy Functions ............................................ 144
    4.3.1 Link Energy Model ..................................................... 144
    4.3.2 Profile Energy Function ............................................. 147
  4.4 Implementations ............................................................. 149
    4.4.1 Corinto, Colombia ...................................................... 149
    4.4.2 Casimiro de Abreu, Brazil ........................................... 157
    4.4.3 Caracas, Venezuela .................................................... 161
    4.4.4 Summary of Results on All Locations ............................... 165
  4.5 Non-location Grouping ..................................................... 167
  4.6 Conclusion & Future Research ............................................. 168

5 Conclusion ................................................................................. 171

A World Cities Data .................................................................. 173
List of Figures

1-1 Growth in the number of Internet users 1995–2016 [71]. .......................... 19
1-2 A screen capture of the profile page for Mayor “Beng” Climaco of Zamboanga City, Philippines [141]. ................................................................. 21
1-3 Example Tweet from the Boston Globe [141]. ................................. 23
1-4 The information hierarchy, taken directly from [144, Figure 1-1, p. 1-3]. ....... 32
2-1 Network search representation as a multi-urn model. .......................... 36
2-2 Venn diagram of the probabilities in the three-urn example. ................. 58
2-3 Comparison of policies $u$, $\tilde{u}$, and $\hat{u}$. ........................................... 68
3-1 ROC curve for the regularized logistic regression classifier for Twitter suspensions ($P_F$ is the false detection rate and $P_D$ is the true detection rate). ............... 90
3-2 Histogram of screen names for active and suspended accounts in our data. The average numbers of screen names for suspended and active accounts are listed in the legend. ................................................................. 94
3-3 Examples of Levenshtein ratios for different pairs of screen names. .............. 95
3-4 Logistic regression ROC curve on hand labeled data ($P_F$ is the false detection rate and $P_D$ is the true detection rate). .................................................. 100
3-5 Graph representation of accounts belonging to the same user using our regression model and equation (3.3) with a threshold of 0.782. ................................. 101
3-6 Paired accounts graph properties as a function of threshold $P$. The threshold value 0.782 from equation (3.3) is indicated on the plot. ................................. 103
3-7 Graph representation of accounts belonging to the same user using our regression model and equation (3.3) with a threshold of 0.668. ................................. 104
3-8 ROC curve for $L_2$-regularized quadratic kernel logistic regression model for predicting re-follows evaluated on out-of-sample test data. (left) Test data and training and validation data can contain the same user. (right) Test data and training and validation data do not contain the same users. ........................................ 109

3-9 Probability of finding the target user’s new account, given it exists, as a function of number of queries of former friend $j$. ................................................................. 119

3-10 Box plots of the percentage above optimal of the (left) expected cost and (right) actual cost of different search polices for ISIS users in Twitter. ...................... 125

4-1 FARC attack locations near Corinto, Colombia since 2002 [104], plotted using Google Maps [64]. ............................................................... 131

4-2 Expand—Classify Methodology ................................................................. 134

4-3 Factor graph model for social media user location classes. ................................. 136

4-4 Energy graph representation of the energy equation corresponding to the factor graph in Figure 4-3. ................................................................. 140

4-5 Illustration of Case 1 cut. Nodes in set $S_L$ are shaded green, while nodes in set $T_L$ are shaded red. Dashed edges are in cut-set $C_L$. ................................. 142

4-6 Illustrations of both minimum cut possibilities for Case 2. Nodes in set $S_L$ are shaded green, while nodes in set $T_L$ are shaded red. Dashed edges are in cut-set $C_L$. ................................. 142

4-7 Illustration of Case 3 cut. Nodes in set $S_L$ are shaded green, while nodes in set $T_L$ are shaded red. Dashed edges are in cut-set $C_L$. ................................. 143

4-8 Decay of link energy $\psi(z_{1,2}, 1, 0)$ as the number of user 1 friends or number of user 2 followers increases. ................................................................. 146

4-9 Corinto, Colombia label radius, plotted on Google Maps [64]. ................................. 150

4-10 Corinto user classification ROC using logistic regression model for profile energy. ................................. 153

4-11 Sensitivity of Corinto user classification to parameter $\gamma$. ................................. 155

4-12 Sensitivity of Corinto user classification to parameter $\lambda$. ................................. 155

4-13 Sensitivity of Corinto user classification to logistic regression regularization. ................................. 156

4-14 Casimiro de Abreu, Brazil label radius, plotted on Google Maps [64]. ................................. 157

4-15 Results and Sensitivity of Casimiro de Abreu user classification. ................................. 160

4-16 Caracas, Venezuela label radius, plotted on Google Maps [64]. ................................. 161

4-17 Caracas, Venezuela Performance. ................................. 162
4-18 Times to complete Caracas collection iterations. . . . . . . . . . . . . . . . . . . . . . 163
4-19 ROC for Pizzagate user classification using hand-labeled test data. . . . . . . . . . 168
List of Tables

1.1 Twitter REST API methods, results, maximum results returned per query (RPQ), and maximum queries per 15-minute interval (Limit) [138]. ......................... 25

3.1 Features for predicting Twitter suspensions. ................................. 89

3.2 Summary of sampled accounts from those incorrectly classified as suspensions using the regularized logistic regression model. ................................. 91

3.3 Partial screen name—tweet timelines for two Twitter user accounts purportedly belonging to Sally Jones. These accounts have been suspended by Twitter and are no longer available. ........................................... 93

3.4 Accounts exhibiting very low similarity, according to the selection criterion given in equation (3.1). ......................................................... 98

3.5 Accounts exhibiting very high similarity, according to selection criterion given in equation (3.2). These accounts were manually labeled as belonging to the same user, i.e., $y^{(i,j)} = 1$. ......................................................... 98

3.6 Regression coefficients for matching accounts. ................................. 99

3.7 Accounts comprising component A. While average hash values for profile pictures are abbreviated, they are the same for all profiles. .............................. 102

3.8 Accounts comprising component B. ................................................. 102

3.9 Accounts comprising component C. ................................................. 105

3.10 Example of @MusabGharieb18’s re-following behavior upon opening new account @MusabGharieb13. The entries in the table indicate who these two accounts followed.106

3.11 Example data rows resulting from re-following behavior given in Table 3.10. Features are omitted but include, for example, characteristics from each friend’s profile. . . . . 106

3.12 Randomly selected account pairs for testing. ................................. 123

3.13 Cost comparisons for different policies. ......................................... 125
4.1 Example odds table used to construct a profile energy function. ............... 148
4.2 List of character strings $W_1$ used to extract profile features for Corinto logistic regression. ............................................................. 151
4.3 Categorization lists for Casimiro de Abreu, Brazil. ................................. 159
4.4 Odds Table Used in Naive Implementations ............................................. 159
4.5 List of character strings $W_1$ used to extract profile features for Caracas logistic regression. ............................................................. 162
4.6 Results from user set collections from nine locations. ............................. 165
Chapter 1

Introduction

In 2013, Philippine journalist Maria Ressa published *From Bin Laden to Facebook* [118], which documented the growing role social media was playing in the growth and reach of extremist organizations. In the 10 years since her first book, *Seeds of Terror* [117], which meticulously documented Al Qaeda’s pre-social media formation and organization in southeast Asia, it seemed that much had changed.

Ms. Ressa was on to something. A year after *From Bin Laden to Facebook* was published, the Islamic State in Iraq and Syria (ISIS) surprised the U.S. and the world with its rapid organization and territorial advances [55, 94]. In addition to its reputation for brutality, ISIS is widely known for its ability to leverage social media for recruiting, propaganda, and communications [13, 61]. The rise of ISIS provides a striking example of the role social media can play in shaping events.

The ubiquity and importance of social media in global events goes beyond ISIS. Access to social media is becoming nearly universal, and much of the content is publicly accessible and available. This content could be an invaluable data source for attaining *situational awareness*, which has its applications in business, military, law enforcement, politics, emergency response, and more. The goal of this research effort is to make meaningful contributions to the growing body of research that aims to attain better understanding, detection, and prediction of events, groups, and societies through the analysis of social media data.

1.1 Problem

The problem we address is to provide new methods of collecting and searching social media data that contribute to improving situational awareness, with a focus on military and security applications.
1.1.1 Technical Approach

This thesis is comprised of three main efforts. In the first effort, we formulate the problem of finding a specific user in a social network based on a probability model assumed on the user’s social connections. We use a dynamic programming approach to formulate the network search and provide necessary and sufficient conditions for optimal search policies in general and in specific cases.

In the second effort, we address the problem of finding Twitter accounts belonging to ISIS users. In this section we use logistic regression models to classify accounts and to provide a probability model on social network connections. We use the results of this probability model and our optimal network search results to find new accounts belonging to previously suspended users.

In the third effort, we address a more general network search problem in which the objective is not to find a specific user, but rather to find all of the users belonging to a specific group or location. We use a factor graph to model the observed social network, and again rely on logistic regression to inform the factor potentials. Efficient optimization is performed using graph cuts in order to find the maximum likelihood classification according to this graph model, with validation and test data used to fit and analyze model parameters.

1.1.2 Experimentation

Because of the relative ease of collecting user generated content, as well as its public availability in general, all of our experimentation in this thesis is carried out on real social media data collected from Twitter [141]. We analyze the performance of our ISIS classification models and our network search model on a dataset consisting of known ISIS members and their social network connections. We analyze our group collection method by applying it to collect Twitter users from nine population centers, and compare the results to those obtained using Twitter’s native search methods.

To evaluate the performance of our classification models, we use the Receiver Operator Characteristic (ROC) curve, which compares the rate of correct positive classifications against the rate of incorrect positive classifications. We use the area under the ROC curve (AUC) as a single metric of performance and to compare different classification models.

1.2 Social Media

In the first decade of the 21st century the nature of the World Wide Web changed. Simultaneously, the number of people connected to the Internet as users (Figure 1-1) began to climb rapidly, the cost
of online data storage declined, and users’ ability to populate sites with their own original content increased [108, p. 2]. The ensuing evolution has come to be known as “Web 2.0,” referring not to any specific technological changes but rather a change in how the World Wide Web was being used [78, p 61]. In contrast to the older “Web 1.0” convention in which a relatively small set of developers created applications and content for users to consume, the environment of Web 2.0 came to be one of collaborative development among users. As a result, the World Wide Web came to be characterized by User Generated Content and Social Media [78, pp 60–61].

![Internet Users Over Time](image)

**Figure 1-1:** Growth in the number of Internet users 1995–2016 [71].

The notion of what constitutes “social media” continues to evolve as new applications are introduced and others are closed or decline in popularity. Boyd and Ellison [21, p. 211] use the term “social network sites” to refer to web based services in which users can create a public (or semi-public) profile, list connections to other users, and view the profiles of their connections and the connections of other users on the site. Kaplan and Haenlein [78, p. 61] provide a simpler definition: “Social media is a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of User Generated Content.” Obar and Wildman [108, p. 2] defines social media services as those having the following features:
1. They are Web 2.0 Internet-based applications.

2. User Generated Content constitutes the application’s “lifeblood.”

3. Users or groups create profiles on the site provided by the social media service.

4. The services enable social networking, so that users can have their profiles connected to those of other users.

The specific implementation of these social media capabilities varies among applications, with the current spectrum of social media sites offering a wide range of diversity in scope, functionality, purpose, and user-base [80, p. 242]. Some sites adapted over time in response to the perceived preferences and specific applications of the users. MySpace, for example, found early success connecting independent rock bands with fans [21, p. 217]. Others, like LinkedIn, were designed to appeal to a specific type of user for a specific purpose. Facebook began as a niche private network for Harvard students, but expanded to include other universities as well, then to high schools, and eventually opened its access to all users to become one of the most frequented social media application in the world [21, p. 218].

Facebook and Twitter, two of the most popular social media applications, are general in nature and widely accessible throughout the world. In both of these applications and in many others like them, individual user identifying information is limited. Facebook allows users to post a substantial amount of identifying information, but users ultimately choose what information to provide and, of what they provide, what information is publicly visible. Twitter allows users to provide a name, a short description, a URL, a picture, and a few other personalizations to their profile [73, p. 59]. Many users of Facebook, Twitter, and other social media use pseudonyms, so that their identities are not immediately apparent.

1.2.1 Twitter

In our experimentation we will use data obtained from Twitter [141], primarily due to the following useful features:

- The Twitter Application Program Interface (API) [138] makes it relatively easy to collect data from this site.

- As a micro-blogging application, use of Twitter centers on real-time updates and conversations in order to create an “ambivalent awareness” of issues [80, p. 244].
• Much of the data is publicly available. While users can “protect” their accounts and their content so that only their “followers” can see it [141], the default setting is for user profiles and User Generated Content to be visible to anybody accessing the site.

User Generated Content in Twitter is restricted to short posts, or *tweets*, by a user. These posts are currently limited to 140 characters, with exceptions or adjustments to this limit when the post includes a URL or attached media. Twitter users can browse other users’ profiles and choose to *follow* specific users. When a user logs on, the Twitter application shows him a *timeline* consisting of recent posts from all of the users he is following, generally sorted in reverse chronological order.

**Creating an Account**

Upon creating an account, a new Twitter user chooses a unique *screen name* that serves as an account login and identifier. The user can also choose a *name* that does not have to be unique, upload a profile picture and a background picture, provide a short *description*, input a *location*, and include a personal *URL*. Twitter assigns each new account a unique, integer *user ID* and stores the date and time of account creation. Figure 1-2 provides an example of an account profile page.

![Figure 1-2: A screen capture of the profile page for Mayor “Beng” Climaco of Zamboanga City, Philippines [141].](image)

**Tweeting**

Once a user has created an account, he can immediately start tweeting. Tweets can include up to 140 characters of text, or can include up to 117 characters of text and a URL hyperlink or attached
photo or video. Twitter assigns each tweet a unique status ID (Twitter refers to tweets as statuses) and stores the time and date it was posted, as well as the author’s user ID. Each tweet is posted with a link to the author’s profile page.

In addition to including URLs and media, the user can mention other users in a Tweet or include hashtags. A hashtag is simply a sequence of numbers and letters preceded by a “hash” symbol (#), which serves as a label on the tweet for the purpose of efficient searching. User mentions are references to other Twitter users by screen name, preceded by the “at” symbol (@). Twitter extracts hashtags and user mentions from tweets and stores them as tweet “entities.” These entities are hyperlinked in the tweet when it is displayed, so that a click on a user mention directs to the mentioned user’s profile page, while a click on a hashtag provides the search results for that hashtag.

Figure 1-3 provides an example of a tweet display. The tweet contains the hashtags “#BREAKING” and “#Bridgegate,” and has two images attached.

Retweeting

A user also has the ability to retweet a tweet from another user. A retweet is similar to a “regular” tweet, except that the retweet contains an explicit reference to the original tweet by beginning with the character string “RT” followed by a user mention citing the original author. Otherwise, the effect of a retweet is the same as that of a tweet: if a user posts a retweet, then that user’s followers will be able to view the retweet on their home timelines. For retweets, Twitter stores the originating tweet’s ID. Twitter also displays the number of retweets with each tweet, which links to the profiles of those that retweeted. As indicated, at the time of the screen capture the tweet in Figure 1-3 has been retweeted 14 times.

Replying

Users can reply directly to tweets. Replies are also treated as tweets, but with special behavior. Replies do not appear on followers’ timelines. Instead, they appear attached to the original, “replied-to” tweet on the original tweet author’s timeline. Replies generally begin with a user mention of the original tweet’s author. Twitter stores replies as tweets with unique tweet IDs and also records the original tweets ID as the in_reply_to_status_id. A single reply can be seen to the tweet in Figure 1-3, although there were more below the screen capture. The reply begins with a user mention of the original tweet’s author but contains another user mention as well. Also, the Twitter web interface provides the authenticating user (me, in Figure 1-3) with the means reply immediately below the
Figure 1-3: Example Tweet from the Boston Globe [141].

tweet.

Liking

A user can indicate interest in a tweet by “liking” the tweet, usually by clicking a button below the tweet. This information is recorded and displayed with the tweet, so that both the number of likes and the screen names of the “likers” can be easily viewed. The tweet in Figure 1-3 had been liked 11 times at the time of screen capture.

Following

If a user is interested in reading content posted by another user, he can follow that user. If the account of interest is protected, then this relationship must be approved by the followed user. Otherwise, consent is assumed (the content is public anyway) and the followed user’s tweets will appear on the follower user’s home timeline.
Searching

Any Twitter user, and even web users without Twitter accounts, can execute search queries through Twitter’s web interface. Twitter returns tweets that match the query criteria. These results can be sorted by most recent, most relevant (as determined by a Twitter algorithm that accounts for how recently the content was posted and how many likes, retweets, and replies it generated), or a mixture of both. Search queries can include hashtags or screen names. Users can also search for accounts, as opposed to tweets, that have names or screen names matching the search criteria.

Surfing Twitter

With the functionality Twitter provides, Twitter users can conduct searches and follow embedded links to related tweets (e.g., by clicking on a hashtag) or user profile pages (by clicking on a user mention or tweet author). In this way the user interacts: replying to and retweeting material the user finds interesting, following other users based on their profile characteristics and their tweets, as well as posting original content.

A user’s profile page provides a concise description of the user. In addition to displaying the relevant information provided by the user (name, screen name, profile and background pictures, description, location, and web URL), the page provides some useful statistics: number of tweets, number of “friends” (accounts the user is following), number of followers, and number of “likes” (tweets the user has liked). Respective links direct a web surfer to displays of all of the user’s friends, followers, or liked tweets. Finally, the user’s profile page enables various options for displaying all of the (unrestricted) content posted by the user.

Figure 1-2 shows the profile page for Mayor Maria Isabella “Beng” Climaco of Zamboanga City, Philippines. Mayor Climaco’s account name, “Beng Climaco”, and screen name, “Beng_Climaco,” are clearly indicated along with the optional description, location, and URL she provided. Much other relevant information is provided as discussed, in a format that is intuitive and easy to understand. The figure also shows that the authenticating user is following Mayor Climaco, so that her tweets will appear on his (my) timeline.

The characteristics and functionality of Twitter as a “micro-blogging” service are designed to enable users to provide real-time updates [80, p. 242]. People use Twitter to share information, report news, keep up with friends, and consume information [73, p. 63]. Observing what information people are sharing, and how it is moving through the social network within the application, could
be useful in attaining situational awareness.

**The Twitter API**

We have mentioned that collecting data from Twitter is relatively easy through the site’s Representation State Transfer (REST) APIs. Queries can be sent to these interfaces, which return the results in an easily readable (e.g. JavaScript Object Notation, or JSON) format. Typical data objects returned by the API methods include *user profile* objects, which contain the user account information (user ID, screen name, name, description, etc.) and statistics (number of tweets, friends, followers, etc.) for a specific user, and *status* or tweet objects, which contain the unique tweet ID, text, posting time, entities, and user profile associated with a specific tweet.

In order to prevent abuse, the various API methods require authentication and have limits on the rate of requests. Table 1.1 provides a summary of some of the more useful queries, the results they return, and their rate limits.

Table 1.1: Twitter REST API methods, results, maximum results returned per query (RPQ), and maximum queries per 15-minute interval (Limit) [138].

<table>
<thead>
<tr>
<th>API Method</th>
<th>Returns</th>
<th>RPQ</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>Tweet objects matching the query criteria</td>
<td>100</td>
<td>180</td>
</tr>
<tr>
<td>Lookup User</td>
<td>User profile objects for queried users</td>
<td>100</td>
<td>900</td>
</tr>
<tr>
<td>Get Friends IDs</td>
<td>User IDs for friends of queried user</td>
<td>5000</td>
<td>15</td>
</tr>
<tr>
<td>Get Followers IDs</td>
<td>User IDs for followers of queried user</td>
<td>5000</td>
<td>15</td>
</tr>
</tbody>
</table>

**1.3 Previous Work**

As social media applications have become pervasive over the past two decades [107, p. 9], research related to social media has also become quite common. Some efforts focus on characterizing and quantifying different aspects of the social networks that grow within these applications. Much research has dealt with how social media has played a role in political and social uprisings. There has also been substantial effort in investigations into how extremist groups, such as the Islamic State in Iraq and Syria (ISIS), have made use of social media as a recruiting and propaganda tool. People have examined how social media can be used to monitor significant events, such as disasters.
or protests, in real time, and many of these and other efforts have aimed to employ social media data also as a predictive tool. In this section we provide a survey of some of the recent research activities dealing with social media and their findings.

### 1.3.1 Social Networks

Sociologists have shown that social networks have played an important role in human societies for centuries [110]. Increases in technology-based connectivity in societies in recent decades has generated increased interest in researching social networks. Some of this work has focused on producing social network generative models that produce networks that are similar to observed social networks. Erdős and Rényi [51] provided a well-known foundation for contemporary network analyses with their random graph model. Many variations, adaptations, and extensions of the random graph model have now been developed. For example, the preferential attachment model [10] has become one of the more popular tools in representing social network formation in recent years. This generative model assumes that new vertices entering the graph connect to higher-degree vertices with higher probability. The authors show that this model naturally leads to a scale-free degree distribution, which closely approximates the empirical degree distributions often found in studied networks.

#### Community Formation Detection

Another aspect of social networks that has been observed empirically is that they tend to be tightly clustered, which has led to the development of several measures of this phenomenon (e.g. average clustering coefficient) [72, p. 34]. This clustering could be at least partially explained by homophily, i.e., the tendency of edges to form between vertices that are similar in some way (e.g., people of the same ethnicity), which has also been extensively documented empirically [50, pp. 77-84]. Backstrom et al. [8] use empirical social network data to show that community formation growth depends on underlying network structure.

However, identifying and characterizing the community structure in a social network is not a well-defined problem, while the building blocks of such a characterization, e.g., finding cliques in a network, are computationally complex operations [53]. Several techniques have emerged to address the community detection problem. Because community detection is an unsupervised task, traditional unsupervised learning methods are often applied. Hierarchical methods, both agglomerative (bottom-up) and divisive (top-down) have been applied, but do not scale well and can be prone to
missing underlying community structures that are not hierarchical in nature [53, p. 93].

Newman and Girvan [106] discuss divisive clustering methods on graphs using various criteria (edge betweenness measures) for ordering the sequence of edge removal. As edges with high betweenness measures are removed from the network, the network begins to separate into separate components, which form the communities. The authors use this process to find the community structure that maximizes graph modularity, a metric they develop to measure the quality of a particular network division in terms of its community strength.

Latent variable methods have also been applied to the community detection problem. Airoldi et al. [3] suggest a mixed membership stochastic block model, and propose an algorithm for learning block (or group) memberships. Membership models have been shown empirically to explain some of the community structure in social networks [24] and have been used in network generative models as well.

One of the canonical examples of community formation in social networks is from Zachary [152], which presents an in-depth analysis of the fracturing of a social network along ideological lines. While dated, this analysis provides a motivating example for our work, as it demonstrates how social network structure and evolution over time can serve as indicators of real world events, ideological group formation, and social upheaval. Our problem is to obtain this information from social media data, which is inherently different than the data used by Zachary to construct his famous Karate Club network.

**Information and Adoption**

Closely related to the study of social network structure is the study of how information and adoption of behavior spreads through a social network. In addition to providing a detailed analysis of the changing structure of the Karate Club, Zachary [152] discusses at length the implications of the structure of the social network on the flow of information.

An interesting structural property concerning the flow of information that has been observed in social networks is what has been termed the “small world” phenomenon, in which randomly selected people (nodes) in very large social networks are typically connected by relatively short paths. Milgram famously studied this property in the 1960s [101, 74]. More recently, Watts and Strogatz [146] and Kleinberg [83] have proposed generative and information models that help explain how social networks might come to exhibit this property and how individuals are often able to use it to transmit a message to another person to whom they have no personal ties.
Other research has focused on the spread of information, ideology, and behavioral adoption, often termed *diffusion*, throughout a social network. A common approach is to apply disease *contagion* models, such as the susceptible-infected (SI) model or the susceptible-infected-recovered (SIR) model to represent how information propagates through a social network [72]. Many adaptations of this fundamental approach have been proposed and studied in the context of diffusion in social networks.

One of the general classes of diffusion models are *linear threshold* models, in which an individual in a network deterministically receives information, or adopts a certain ideological position or behavior, once the fraction of the individual’s neighbors that have adopted the behavior exceeds a given threshold [66].

Another class of models takes a probabilistic approach, in which an individual becomes infected in each discrete stage according to a probability that depends on the number of infected neighbors the individual has [62]. Gomez Rodriguez et al. [63] use this approach to infer a social network based on observations of information diffusion or adoption behavior.

Dow et al. [46] provide an empirical study of how a small proportion of user-generated content propagates quickly over large social networks, or “goes viral,” while most does not spread beyond a small subgraph. Another study by Yang et al. [148] employs semi-supervised learning with some success to predict how users will propagate information by “retweeting” other users’ content on Twitter. Cheng et al. [32] show that the “shape” of the overall diffusion of a topic in social media can be predicted based on characteristics of its initial spread.

**Influence**

In the course of studying diffusion, it is natural to also investigate which people within a social network are the most *influential*, i.e., would have the most impact on propagation of information or adoption throughout the network. Because finding influential people has many applications, such as marketing and politics, much work has been done in this area. Kempe et al. [79] investigate the problem of determining the most influential nodes in a social network using both the context of linear threshold models and probabilistic cascade models of diffusion. Pei et al. [112] provide an empirical look at different measures of influence in the context of information diffusion, finding in some cases that the underlying graph structure can be misleading. Shah and Zaman [128] introduce a metric for detecting an information source given propagation information.
1.3.2 Social Media

The transition to Web 2.0 and the rise of social media have provided rich sources of data that is difficult to concisely characterize. Embedded in social media applications are network structures: the ability of users to connect to one another has been proposed as a defining characteristic of social media [108, p. 2]. Much research has treated these networks as observable social networks, but not all agree. Couldry and van Dijck [36] provide an interesting principle-based argument for why social media is not at all “social” in a traditional sense. Ariel and Avidar [6] conduct a similar inquiry with a less pessimistic approach, and conclude by admonishing researchers to always investigate the interactive social nature of social media applications, rather than assuming them. Furthermore, structural phenomena such as “condensation,” in which a single “superstar” vertex obtains a positive fraction of the edges in a graph, have been observed in online social media networks [17] but do not easily translate to real-world social networks.

Situational Awareness and Understanding

Many varying applications of enhancing situational awareness using social media, and especially Twitter, have been investigated. Examples include tracking earthquakes [37, 124], visualizing events through geotagged photos [103], tracking the spread of infectious diseases [29, 102], and monitoring natural disasters, emergencies, and emergency relief operations [145, 116, 150, 65].

Other studies have focused on more subtle elements of situational awareness. For example, Java et al. [73] analyze why people use Twitter and investigate underlying community structures. The authors assert that this information could be beneficial to Twitter in developing new features and retaining users. In this case, user-generated content is analyzed to infer invisible characteristics of a population, rather than visible events and activities.

The use of social media data and analyses to attain situational awareness have led to ethical implications in some cases. Lory [95] investigates the ethical implications of using Facebook accounts to recruit and evaluate prospective job applicants. In some cases, employers were even asking candidates for their social media passwords, a practice that led to many states enacting laws banning such practices [39]. In mid 2016, Twitter suspended U.S. Intelligence agencies’ access to data collected by Dataminr, Inc., because the social media provider did not want its data used for government surveillance purposes [133]. Even more recently, Facebook and Twitter cut off data access to Geofeedia, a firm that inferred user location based on user-generated content and network
structure, citing similar concerns [135]. Some work has been done in investigating the ethics and legality of these uses of social media [57], while governmental organizations have been developing official social media policies (e.g., [109]). These cases serve as a caution on the ethical implications of our work, which must be careful to maintain privacy and adhere to social media site terms of use.

We now review some of the relevant literature on social media in specific contexts relevant to our problem. We include research explicitly focused on deriving situational awareness from social media, as well as investigations into the role social media plays in shaping real-world events. We include research in the latter category because it could provide additional insights into how social media could be of use in enhancing situational awareness.

**Political Polarization, Unrest, and Conflict**

Even before the transition to Web 2.0, several studies had observed that, given the freedom to do so, people tended to self-segregate into homogeneous groups (e.g., see [18]). Rosenau [121, p. 17] and others have suggested that the unprecedented connectivity enabled by new technologies have increased the fragmentation and polarization in society, making it easier than ever for a person to connect with and mobilize other like-minded people.

Nissen [107] provides an in-depth analysis of the role social media has played in armed conflicts since the 1999 Kosovo War. Several authors have analyzed how social media was used to catalyze the 2011 Egypt Revolution and the Arab Spring uprisings [81, 131, 49, 4, 126, 42, 28]. Others have examined the utility of using social media to track ongoing conflicts. Carley et al. [30] outline a procedure for obtaining real-time information on developing crises using Twitter and open source media, using the 2012 attack on the U.S. Consulate in Benghazi, Libya as a case study. Starbird et al. [132] employ a collaborative filtering technique to identify social media users who are physically present during mass disruptions.

Other work focuses on using social media as a means to track and quantify underlying controversies and conflicts within a population. Marres and Moats [98] give an assessment of the usefulness of social media in analyzing controversy. Romero et al. [120] explore how controversial topics have different diffusion patterns than non-controversial topics in Twitter. Many case studies have looked at how specific causes have evolved on social media, e.g., [70]. Seo et al. [127] use community detection on topics to determine how polarized a population is, using tweets collected during the 2011 Egypt Revolution as a case study.
In addition to studies on how social media can be used to influence, identify, and track controversy, unrest, and conflict, much work has investigated social media’s potential in law enforcement and terrorism applications. Nauta [105] proposes a method for using Twitter to locate and track a person of interest, Peters [114] suggests using social media as a nation-wide reporting system for suspicious activity, and Fresenko [54] analyzes the utility and challenges associated with standing up government-operated social media analysis centers for law enforcement purposes.

The rise of the Islamic State in Iraq and Syria (ISIS) has directed much attention at the role social media can play in both spreading and combating terrorism. Ressa [118] alludes terrorist groups’ use of social media prior to the formation of ISIS. Berger [13] and Klausen [82] provide empirical analyses of ISIS’s use of Twitter to recruit, garner support, propagandize, and intimidate. Alpern and Lidbetter [5], and Benigni and Carley [11], provide methodologies for finding, detecting, and monitoring online terrorists. Meanwhile, social media sites such as Twitter continue to suspend accounts that promote violence and extremism [1].

### 1.4 Situational Awareness and Intelligence

Our research goal is to exploit social media data in ways that are useful in the context of situational awareness. In this section we provide a very brief overview of situational awareness in the context of military operations.

The notion of situational awareness, sometimes referred to as situational understanding, is ubiquitous in military operations, but has applications in politics, emergency response, and many other areas. Describing the operational environment is accomplished through military intelligence [143, p IV-5], with the primary objective of reducing uncertainty by “providing accurate, timely, and relevant knowledge about the threat and surrounding environment.” [144, p. 1-1]

The military defines the operational environment as “the composite of the conditions, circumstances, and influences that affect the employment of capabilities and bear on the decisions of the commander.” U.S. military intelligence doctrine applies a “holistic” view to understanding the operational environment that encompasses physical areas and factors and the information environment [142, p. I-2].

Achieving situational awareness is more than just possessing data or information [144, p. 1-3]. Figure 1-4 provides a depiction of the information hierarchy, and suggests a process by which
data and information are collection, analyzed, and transformed into situational awareness that is ultimately used in decision making. In the military, this process is known as the intelligence cycle [144, p. 1-3] or intelligence process [142, p I-5]. Our intent is to contribute methods that enable the processing of social media data into useful intelligence.

![Information Hierarchy Diagram](image)

Figure 1-4: The information hierarchy, taken directly from [144, Figure 1-1, p. 1-3].

### 1.5 Thesis Organization

Chapter 2 presents the formulation of the social network search problem, described in the context of drawing marbles from urns. This chapter provides a derivation of general necessary conditions for optimal search policies as well as a complete characterization of the optimal policy in two specific cases. Technical proofs of our optimality conditions and other findings are included at the end of this chapter.

Chapter 3 introduces the ISIS dataset and provides multiple analyses on this data. These analyses address several problems: identification of extremist accounts within the data, identification of multiple accounts belonging to a single user, prediction of individual user networking behavior...
when opening a new account, and finding a new Twitter account belonging to a specific user. This final effort applies the optimal search policies found in Chapter 2, and includes analysis of performance using the ISIS dataset.

Chapter 4 presents a collection methodology and factor graph classification approach to obtaining a set of users belonging to a specific group or location. Collections of users from nine population centers and one ideological group are performed and evaluated.

Finally, Chapter 5 provides some final comments on future work and applications, concluding this thesis.
Chapter 2

Optimal User Search in Online Social Networks

In this chapter, we analyze the problem of searching a social network for a particular target individual by sequentially examining the neighbors of other known users. Social media applications enable users to connect with each other, forming social networks. For different reasons which we will discuss, one may wish to find a target individual in the social network. One may have prior knowledge that the target individual is connected with a set of known users, and so the most logical place to begin searching is the neighbors of these known users. If querying each of these neighbors incurs some sort of cost, then the goal would be to find the target with as few queries as possible.

2.1 Background and Motivation

Suppose Mary is searching a social media application for an account belonging to John, an old friend from school with whom she has lost contact. From what she knows about John, Mary might be able to develop a list of accounts she knows about within the social media application to which John’s account might be connected. For example, she might recall that John was good friends with Matt, who has a social media account that is known to Mary. She also might remember John was active in a certain charity, which also maintains a social media account known to Mary.

After developing such a list, Mary could sequentially explore each account’s connections, but doing so could take a substantial amount of time. In order to find John’s account quickly (assuming John has an account), Mary might devise a search strategy. For example, she might start by looking at accounts she feels are the most likely to be connected with John’s account. Alternatively, she
might start by looking at accounts with fewer connections, because her goal is to find John’s account while minimizing the time spent exploring.

In this hypothetical scenario, what Mary is doing is an example of a network vertex search in which the sought object, or target might be found by examining the neighbors of a finite set of known vertices. Once the search target is found, the search typically terminates. Each known vertex $i$ might have a different degree, $N_i$, requiring a different number of search queries to exhaustively search. From our scenario, we also consider that each known vertex $i$ might have a different probability of being connected to the target vertex, which we denote as $\varphi_i$.

This scenario motivates our probabilistic “multi-urn” search model for finding users in social networks, in which we represent each known vertex as an urn containing a finite number of marbles. Each marble in an urn represents one of the respective vertex’s neighbors. The search consists of successively drawing and examining individual marbles from the urns with the goal of finding a red marble, representing the search target, in the fewest number of draws. Figure 2-1 depicts the multi-urn model for a network search on three known vertices. In this example, each of the known vertices is connected to the search target, so each urn contains a single red marble. Additionally, each urn contains a number of blue marbles that represent the other neighbors belonging to each respective vertex. Employing a dynamic programming framework, we provide some insight into the optimal search policy in this general model and give an explicit characterization of the optimal policy under certain conditions.

![Figure 2-1: Network search representation as a multi-urn model.](image)
2.1.1 Application

Our model applies to any scenario where one must sequentially search for the target amongst a set of entities which are separated into different clusters. In our vertex search problem, the entities are vertices and the clusters are neighborhoods of the known vertices. Our main motivation for this model is in social media applications where many times the goal is to find users who harass others, incite violence, or engage in other dangerous behaviors. Twitter has been suspending large numbers of users, many of which support or engage in violent extremism, from its micro-blogging application for violating the site's published rules [67]. The challenge is that these users can simply create a new account each time one is suspended. However, from historical data Twitter could predict the accounts to which the suspended user is likely to connect. Using this information, Twitter administrators could then apply our optimality criteria to efficiently locate new accounts belonging to suspended users.

There are other scenarios where this model can apply. For instance, for law enforcement and intelligence applications, the search entities could be suspects in a crime and the clusters could be geographical locations. Or if one is examining a dump of emails from a suspect’s server, one may be looking for an incriminating email, so the entities are emails and the clusters could be recipients of the emails. In both of these examples, the process of querying the entities requires a non-trivial amount of resources (interviewing a suspect, reading an email), so it is important to find the target as quickly as possible. Using an optimal or near-optimal search strategy is therefore crucial in these examples.

2.1.2 Previous Work

Much of the work that has been done in the context of network search is focused on finding relevant vertices in a large scale network. Google’s PageRank algorithm is probably the most well known example of these methods, of which many adaptations and generalizations exist [15]. Our work looks at an essentially different type of network search: one of finding a specific vertex in a network, presumably identifiable by certain features, by investigating network neighborhoods in which the vertex is likely to appear.

Our network vertex search problem is more closely related to classical search problems in operations research. Black [19] poses a simple search problem in which the searcher can look in a finite number of locations, each of which has a search cost, a probability of containing the target, and a
known “miss” probability of not finding the target upon searching the location, given the target is there. These probabilities are assumed to be stationary.

Black’s method and findings somewhat parallel ours, although our network vertex search requires substantially different assumptions. For one, the probability of finding the target vertex neighboring a certain known vertex changes with the number of times it has been searched, and the known vertex’s neighbors can eventually be exhausted. Also, the target can be found neighboring more than one of the known vertices in the network, and therefore might be found in multiple “locations” in our network vertex search.

Physical network search problems are also addressed in the literature. These typically differ from Black’s search model, as well as from the network vertex search we propose, by applying network flow constraints to the searcher and the target. Dagan and Gal [38] provide such a model in which the search target is assumed to be “hiding” at a stationary point (not necessarily at a vertex) in a network. The searcher in this problem chooses his starting point, and then choses a search path with the objective of minimizing expected time to find the target. This model is related to the well-known Chinese Postman Problem, which involves finding the minimal cost path that traverses every edge in a network.

[93] considers a more general search model that consists of \( k \) balls (targets) hidden among \( m \) boxes, an approach that closely resembles our development. In this approach, the objective is to search the boxes in an order that minimizes the expected cost of finding all \( k \) distinct objects. In addition to providing relevant theoretical results on optimality under their modeling assumptions, the authors show that in this search problem, as in ours, there is no benefit to making decisions based on previous search results. The authors also relate this problem to other work on physical network search, specifically their findings on expanding searches [5].

Similar to the early model proposed by Black [19], the network vertex search problem we propose has many of the characteristics of the well-studied multi-arm bandit problem. Bubeck and Cesa-Bianchi [27] provide a broad survey of many variations of the multi-arm bandit problem and their respective applications. These problems are typically likened to a gambler who has a choice of playing from a set of slot machines. At each discrete stage in the process the gambler selects and plays a slot machine for a certain cost and receives a stochastic reward from an unknown distribution. The more times the gambler plays a particular machine, the more he is able to learn about its reward distribution.

In the multi-arm bandit setting, the gambler would not want to spend too much money playing
low-payout slot machines just to learn their reward distribution. This quandary is the fundamental trade-off between exploration and exploitation, which is inherent in multi-arm bandit problems. In order to make money, the gambler wants to play only the highest-payout slot machine. However, he never really knows the true distributions of any of the machines. As a result, optimal multi-arm bandit policies often include a balance of exploratory actions, in which decisions are made for the sole purpose of observing outcomes, and exploitative actions, in which decisions are made to optimize the outcomes based on what has been learned.

The multi-arm bandit problem objective is often characterized as the minimization of regret, which is essentially the difference in expectation between what the gambler earns and what he would have earned by playing the best machine. Lai and Robbins [88] provide a very well-known method for constructing adaptive multi-arm bandit policies using upper confidence bounds, for which regret grows proportional to the logarithm of the number of plays in the limit. Auer et al. [7] show that this same bound on regret is also achievable in finite time.

The multi-urn search model we present could be cast in the context of a finite time multi-arm bandit problem, but there are a few notable differences. Our objective, to find the search target as quickly as possible, does not immediately cast itself as minimizing regret. Gittins [60] overcomes this difficulty by augmenting the state space in the multi-arm bandit formulation with a “success” state, from which no additional costs or rewards are incurred. Building on this adaptation, Gittins describes a class of search problems that are very similar to our network search problem, and characterizes the optimal search policy based on his well-known dynamic allocation index [59].

Our search problem differs from that of Gittins [60], however, in that each vertex has a fixed, finite, and known number of neighbors. In essence, we assume the reward distribution of each slot machine is known, and we only allow a fixed, finite number of plays on each machine. Unlike the bandit approach, the outcomes of successive marble draws from a single urn are not assumed to be independent observations from an unknown distribution. Instead, our model uses a known distribution on each machine but limits each machine to allowing at most a single win. Furthermore, in our approach we allow for dependencies between the urns, whereas the multi-arm bandit approach typically assumes each slot machine’s outcomes are independent of the others.

In spite of these differences, the dynamic allocation index applied in the class of search models proposed by Gittins [60] has many similarities to our development. The system dynamics in both cases are governed by Bayesian probability updates. We show that in at least two cases the optimal policy can be characterized by a priority index, which is derived directly from the system
dynamics and can be interpreted as the expected rewards of decisions. Gittins [60] also mentions monotonicity properties of his dynamic allocation index in the context of search that are similar to the monotonicity properties we derive. Our method for proving optimality uses similar logic to the proofs given by Frostig and Weiss [56], which are based on the original development by Gittins and Jones.

Like multi-arm bandit problems, urn models have been applied in many contexts, including discrete decision processes. The Pólya urn process is a well-known construct using urns that has been adapted and used in many applications [96]. This process generally consists of one or more urns, each containing certain numbers of marbles of different colors. At each stage in the process a marble is randomly drawn from an urn and its color observed. This color then dictates an action involving placing one or more marbles of certain colors into certain urns.

Wei [147] provides a specific adaptation the Pólya urn process to the problem of conducting medical trials in a way that is meant to exploit the use of treatments that have shown positive results in the past, which is very similar to multi-arm bandit models applied in the same context. The Pólya urn process has also been used as a preferential attachment model in the formation of networks [34]. This application can be useful in considering how links form in social networks, and is similar to our problem. We assume, however, that the links are already present in the network and are instead interested in finding the optimal way to investigate these existing links.

Downey et al. [47] employ a multi-urn model that is very similar to ours but serves a different purpose: unsupervised information extraction. The model these authors propose uses urns to represent different collections of documents. Marbles drawn from the urns represent specific documents, from which specific labels are extracted. The objective of the model is to learn which labels are the correct, or “target” labels, and which labels are erroneous extractions.

The urn model proposed by Downey et al. [47] differs substantially from ours in its objective. Downey et al. have the objective of learning model parameters and, in the unsupervised case, learning which labels are correct. In the urn model we present, we assume the probability distributions and the target labels are known a priori, and we aim to to find a target marble as efficiently as possible.

Our network search problem is also related to the problem of mutual information maximization. If our goal was mutual information maximization, we would not necessarily focus our search effort on trying to find the target vertex. Instead, we would examine the places that would give us the most information about where the target is likely to be. This is similar to the goal of exploration in the
multi-arm bandit problem. Chen et al. [31] analyze a sequential information maximization problem that parallels our development, using a dynamic programming approach and giving bounds on the performance of the greedy approach. The problem the authors propose involves learning about the distribution of an unknown parameter of interest by sequentially observing other variables. Each observation provides some information about the unknown parameter, and the objective is to maximize the total information gained in a fixed number of observations.

Our multi-urn search model departs most substantially from the development of Chen et al. [31] by imposing additional constraints and dynamics in the way observations are made. In our model, the urns are depleted over time, changing the amount of information contained in each successive marble drawn in predictable, but sometimes unintuitive ways. Our main contributions in this chapter are the characterizations of optimal search policies under various probability models, which come directly from analysis of the dynamics inherent in our multi-urn search model.

Finally, recent work in scheduling and inspection policies employ similar dynamic programming approaches to characterize optimal policies. Levi et al. [91] use dynamic programming to find policies that optimally allocate resources between information gathering and task execution. This class of models provides a natural extension to our network search problem. While we assume a probability model on a set of known vertices, using this approach we could attempt to find the optimal balance between the time spent learning a probability model on a set of known vertices and the time spent executing the search on the current known vertex set.

### 2.2 Multi-urn Search Model

We return to the context of network vertex search as presented in the introduction. We make the following assumptions:

- We have a known set of users, or vertices in the social network, which might be connected to the target user. Let $\mathcal{V}$ be this set of vertices.

- We know the number of connections each user in set $\mathcal{V}$ has. In other words, we know the degree of each vertex $i \in \mathcal{V}$. Let $N_i$ be the degree of user $i$. This is the number of search queries required to exhaustively search through all of $i$’s connections.

- There is a fixed cost of one query associated with examining a neighbor of any user $i \in \mathcal{V}$. 
If we have not executed a query to examine a particular neighbor of user \( i \in \mathcal{V} \), then we have no information about that neighbor.

The neighbors of each vertex are queried in a random order, so that each individual neighbor query of a particular vertex is equally likely to be the search target, given that the target is connected to the queried vertex.

We have a probability model that quantifies our belief that our search target is connected to each user \( i \in \mathcal{V} \).

The target user’s account is immediately identifiable to the searcher, i.e., there is no uncertainty when examining a user account as to whether or not the account belongs to the target user.

Under these assumptions, we can represent this search problem as an experiment involving randomly drawing marbles from a set of urns, where each urn represents a known vertex in the network. Each marble in urn \( i \in \mathcal{V} \) represents a neighbor of vertex \( i \). The degree of vertex \( i \) is \( N_i \), so urn \( i \) initially has \( N_i \) marbles. With probability \( \varphi_i > 0 \), exactly one of the \( N_i \) marbles in urn \( i \) is red, indicating that known vertex \( i \) is connected to the target vertex. Otherwise, all marbles in all urns are blue.

Querying a random neighbor of vertex \( i \) in search of the target is analogous to drawing a random marble from urn \( i \) and observing its color. If the marble is red, the target vertex has been located. If the marble is blue, the target has not been found and the search continues with the remaining marbles. Note that blue marbles are not put back into the urns; once they are drawn they are discarded. Just as Mary desires to find her old friend John with as few searches as possible, the goal in this experiment is to minimize the number of blue marbles drawn, or neighbor queries executed, before finding the target.

We now more completely specify the probability model that accounts for how the target vertex might be connected to the set of known vertices, i.e., how red marbles might be distributed among the urns. Let \( A_i \) be the event that the target vertex is connected to vertex \( i \). We have already defined

\[
\varphi_i = \mathbb{P}(A_i).
\]

More generally, we let

\[
\varphi_U = \mathbb{P}\left( \bigcap_{i \in U} A_i \right)
\]
be the probability that the target vertex is connected to all vertices in set $U \subseteq \mathcal{V}$. If we were to assume that the target would connect to the members of $U$ independently, then $\varphi_U = \prod_{i \in U} \varphi_i$.

In general, the connections might not be independent. For example, Mary might think that if John connected with a certain musician he liked, he might be more likely to connect to other, similar musicians. In other cases, a connection to a particular vertex might imply a decrease in the probability of connection to another vertex.

In our urn model, we assume a known probability $\varphi_U$ for all subsets $\{U : U \subseteq \mathcal{V}\}$, which fully specifies a probability model on the locations of the red marbles among the urns. It allows for arbitrary correlations between urns, so that the presence of a red marble in one urn (or subset of urns) can have a positive or negative correlation with the presence of a red marble in another urn (or another subset of urns).

We note now that the empty set $\emptyset \in \{U : U \subseteq \mathcal{V}\}$. By convention, we set $\bigcap_{i \in \emptyset} A_i = \Omega$, so that $\varphi_{\emptyset} = 1$. This term is implicitly included in summations over all subsets expressed in our development. For example, the summation

$$\sum_{U \subseteq \mathcal{V}} (-1)^{|U|} \varphi_U$$

includes a “1” corresponding to the case in which $U = \emptyset$.

Given this set of probabilities, the probability of any specific outcome of marble locations, or vertex connections, can be determined using the well-known inclusion-exclusion formula. For example, suppose we are interested in the probability that the marble is located in all of the urns in set $U$ and no other urns. This event can be written as $\left(\bigcap_{i \in U} A_i\right) \cap \left(\bigcap_{j \in \mathcal{V} \setminus U} A_j^c\right)$, with

$$\mathbb{P}\left(\left(\bigcap_{i \in U} A_i\right) \cap \left(\bigcap_{j \in \mathcal{V} \setminus U} A_j^c\right)\right) = \sum_{S : S \subseteq \mathcal{V}, U \subseteq S} (-1)^{|S|-|U|} \varphi_S \geq 0. \quad (2.1)$$

We refer to the type of search described in this section as a multi-urn search problem which we now more formally define.

**Definition 1.** A multi-urn search problem is a search problem that can be modeled as sequentially drawing marbles from a set of urns, $\mathcal{V}$, where

1. The objective of the searcher is to find a red marble with as few draws as possible.

2. Each urn $i \in \mathcal{V}$ contains at most a single red marble. Otherwise, all marbles are blue.
3. Each urn $i \in \mathcal{V}$ contains a known number of marbles, $N_i$.

4. For each subset of urns $U \subseteq \mathcal{V}$, the probability that a red marble is present in all urns in $U$ is $\varphi_U$. Additionally, we set $\varphi_\emptyset = 1$.

The network vertex search problem we used to motivate this model can be characterized as a multi-urn search problem, but this model might have other useful applications as well. For this reason, in the development that follows will make use of our multi-urn representation, using the language of “urns” and “marbles,” though we could immediately recover our original context by substituting “known vertices” and “neighbors,” respectively.

2.2.1 Dynamic Programming Framework

In the search model we have defined, the decisions are carried out sequentially in discrete stages. We now take a dynamic programming approach [16] to framing this problem.

We model the search process as a discrete dynamic system of the form

$$x(t + 1) = f(x(t), u(t), w(x(t), u(t))),$$

where $t = 0, 1, \ldots$ is the stage of the search, which we equate to the total number of marbles already drawn from the urns. The system state, $x(t)$, is a record of the total number of marbles drawn from each urn, which sufficiently characterizes the system at stage $t$. Parameter $u(t)$ is the decision made, or urn selected, at stage $t$, and $w(x(t), u(t))$ is a binary stochastic input that indicates whether a red marble is drawn from urn $u(t) \in \mathcal{V}$ in state $x(t)$.

If a red marble is drawn at stage $t$, then $w(x(t), u(t)) = 1$ and the search terminates. Otherwise, $w(x(t), u(t)) = 0$ and the search continues. Letting $x_i(t)$ be the number of marbles that have been removed from urn $i$ at time $t$, we can explicitly define the state vector

$$x(t) = (x_1(t), x_2(t), \ldots, x_{|\mathcal{V}|}).$$

If a blue marble is drawn from urn $u(t)$ in state $x(t)$, the state transition function is:

$$f(x(t), u(t), 0) = x(t) + e_{u(t)},$$

where $e_i$ is the $i$th unit vector. If a red marble is drawn at any stage, the search terminates.
Our dynamic programming model consists of at most \( N + 1 \) stages, where \( N = \sum_{i \in \mathcal{V}} N_i \) is the total number of marbles summed over all of the urns, and is finite.

We define a valid policy \( u \) as a sequence of decisions \((u(0), u(1), \ldots, u(N - 1))\), where \( u(t) \in \mathcal{V} \) for \( t = 0, 1, \ldots, N - 1 \), and for which

\[
|\{t : u(t) = i\}| = N_i \quad \forall i \in \mathcal{V}.
\]

This final condition ensures that the policy will eventually exhaust each urn, as long as a red marble is not found, while at the same time never attempting to draw marbles from an empty urn. A searcher executing a valid policy draws a marble from urn \( u(t) \) at each stage \( t \) until either the target marble is found or there are no marbles remaining in any of the urns, in which case the entire policy has been executed.

We note that in this dynamic programming model there is no benefit in making policy decisions during search execution. At each stage the searcher either draws a red marble, in which case she stops looking, or draws a blue marble and keeps searching. A valid policy provides an ordering of urn queries that is essentially conditioned on not drawing a red marble, which can be considered a deterministic process governed by our simple state transition function. The expected search outcomes for such a policy can be analyzed and compared to those of other policies a priori.

**Stage \( t \) Probability of Drawing a Red Marble**

Building on our dynamic programming modeling assumptions, we now develop the probability distribution associated with \( w(x(t), u(t)) \). Recall that this function indicates whether a red marble is drawn in stage \( t \): \( w(x(t), u(t)) = 1 \) implies a red marble is drawn from urn \( u(t) \) at stage \( t \), while \( w(x(t), u(t)) = 0 \) implies a blue marble is drawn from \( u(t) \) at stage \( t \).

In determining the probability distribution of \( w(x(t), u(t)) \), it is important to remember that in order to arrive in stage \( t \) while executing policy \( u \), the preceding queries \( u(0), u(1), \ldots, u(t - 1) \) would have been executed without drawing a red marble, so that the system arrives in state \( x(t) \). For simplicity of notation, we condition an event on state \( x(t) \) to imply that state \( x(t) \) has been reached without having drawn a red marble. For example, \( P(A_i | x(t)) \) represents the probability urn \( i \) contains a red marble, given queries \( u(0), u(1), \ldots, u(t - 1) \) have been executed without drawing a red marble.
Using the multiplication rule, we can write the probability
\[
P(w(x(t), u(t)) = 1) = \left(\frac{1}{N_u(t) - x_u(t)}\right)P(A_u(t) \mid x(t)),
\]
which is the probability of drawing a red marble from the \(N_u(t) - x_u(t)\) marbles remaining in urn \(u(t)\), given there is a red marble in \(u(t)\), multiplied by the probability urn \(u(t)\) contains a red marble given the system has arrived at state \(x(t)\).

The complementary probability can be written using the law of total probability:
\[
P(w(x(t), u(t)) = 0) = \left(1 - \frac{1}{N_u(t) - x_u(t)}\right)P(A_u(t) \mid x(t)) + P(A^c_u(t) \mid x(t)) = 1 - \left(\frac{1}{N_u(t) - x_u(t)}\right)P(A_u(t) \mid x(t))
\]

Stage \(t\) Urn Probabilities

In this process, we have assumed a fully specified initial probability model on the urns, i.e., for any subset \(U \subseteq \mathcal{V}\), the probability that a red marble is present in all of the urns, \(\varphi_U\), is known. This probability model can be thought of as a Bayesian prior, a quantification of the searcher’s beliefs on where a red marble might be found.

However, these probabilities are not static. After drawing a marble from an urn, the probabilities change as a result of the new information. If the marble drawn is red, then the probability that a red marble existed in the queried urn becomes 1. Likewise, if the marble drawn is blue, then the probability that a red marble can be found in the queried urn decreases as a function of the number of marbles remaining in the urn and the current urn probability.

As long as a red marble is not found, the evolution of urn probabilities over the course of the search is completely determined by the initial probability model and the search policy. We now provide a general expression for updated urn probabilities at stage \(t\).

**Theorem 1** (Urn Probabilities). In a multi-urn search problem over a set of urns \(\mathcal{V}\), suppose a red marble is not found in the first \(t\) queries when executing a valid policy \(u = (u(0), \ldots, u(t-1))\). Then, for any subset of urns \(U \subseteq \mathcal{V}\), the probability of a red marble being in all of the urns in \(U\) at stage \(t\) is given by:
\[
P\left(\bigcap_{i \in U} A_i \mid x(t)\right) = \frac{\prod_{i \in U} \left(1 - \frac{x_i(t)}{N_i}\right) \sum_{\{S \subseteq \mathcal{V} : S \supseteq U\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j}}{\sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j}}.
\]
The proof of Theorem 1 is in Section 2.6, at the end of this chapter, at the end of this chapter.

Substituting the result in Theorem 1 into Equations (2.2) and (2.3) gives us the following corollary.

**Corollary 1.** The probability distribution of \( w(x(t), u(t)) \), indicating whether a red marble is drawn at stage \( t \) is,

\[
\mathbb{P}(w(x(t), u(t)) = k) = \begin{cases} 
\frac{\sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{i \in S} x_i(t+1)/N_i}{\sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{j \in S} x_j(t)/N_j}, & k = 0 \\
\frac{1}{N_{u(t)}} \frac{\sum_{S \subseteq \mathcal{V}: u(t) \in S} (-1)^{|S|} \varphi_S \prod_{i \in S \setminus \{u(t)\}} x_i(t)/N_i}{\sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{i \in S} x_i(t)/N_i}, & k = 1.
\end{cases}
\]

Theorem 1 provides a few important insights. First, we can see that the urn probabilities at any stage depend only on the numbers of marbles drawn from all of the urns, and not the order in which they were drawn. This *path-independence* property of the stage \( t \) urn probabilities is somewhat intuitive: a specific set of marbles drawn gives us a fixed amount of information irrespective of the order in which we inspect the marbles. We will make use of the path-independence property in the proofs for Theorems 2, 4, and 6.

Another observation is that the form of the probability expression is similar to the well-known inclusion-exclusion formula for computing probabilities of unions of events. In fact, this probability is an application of the principle of inclusion-exclusion applied in conjunction with Bayesian updates. In Lemma 1, provided in Section 2.6, we explicitly define the events that are characterized by the inclusion-exclusion formulas in Theorem 1.

### 2.2.2 Costs

In many network search applications, the cost of finding and examining a (random) neighbor of a known vertex is primarily the time consumed in executing the query and reviewing the results to determine whether the neighbor is the search target. Because we have no reason to believe this time-cost would be different for different vertex-neighbor queries, we assume in our model that the cost of drawing a marble is the same for all urns. The goal of the searcher is simply to minimize the number of blue marbles drawn, or vertex-neighbor queries executed, before finding the search target.
We therefore define the cost function at stage \( t \),

\[
g(t) = \begin{cases} 
1 & w(x(t), u(t)) = 0 \\
0 & \text{otherwise}, 
\end{cases}
\]

which applies a unit cost for every blue marble drawn. Because this quantity is stochastic, we set as our objective the minimization of expected total cost. Letting random variable \( C = \sum_{t=0}^{N} g(t) \), we aim to find the optimal policy \( u \) to solve the following optimization problem:

\[
\min_{u} \mathbb{E}[C].
\]

Because \( C \in \{0, 1, \ldots, N\} \) almost surely, we can write

\[
\mathbb{E}[C] = \sum_{k=0}^{N-1} \mathbb{P}(C > k) \\
= \sum_{k=0}^{N-1} \prod_{t=0}^{k} \mathbb{P}(w(x(t), u(t)) = 0). 
\tag{2.4}
\]

The product in this summation, \( \prod_{t=0}^{k} \mathbb{P}(w(x(t), u(t)) = 0) \), is exactly the probability of making it to stage \( k + 1 \) without having found a red marble. From Corollary 1 we can find an expression for this probability.

**Corollary 2.** Given a multi-urn search problem and valid policy \( u \), the probability of arriving in stage \( k + 1 \) without having found a red marble is

\[
\mathbb{P}(C > k) = \prod_{t=0}^{k} \mathbb{P}(w(x(t), u(t)) = 0) \\
= \sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{i \in S} \frac{x_{i}(k + 1)}{N_i}. 
\]

We can therefore rewrite the cost function,

\[
\mathbb{E}[C] = \sum_{t=0}^{N-1} \sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{i \in S} \frac{x_{i}(t + 1)}{N_i}. 
\tag{2.5}
\]

Substituting the probability from Corollary 2 into Corollary 1 reveals an interesting property of the
dynamics of this system:

\[ P (w(x(t), u(t)) = 0) = \frac{P(C > t)}{P(C > t - 1)}. \] (2.6)

### 2.3 Key Results

The cost function in equation (2.5) is nonlinear and nonconvex. Additionally, for a solution to be feasible, the values for \( x_i(t), i = 1, \ldots, |\mathcal{V}|, t = 0, \ldots, N \) must be constrained to correspond to stages reached by a valid policy. Nonlinear, non-convex constrained optimization is difficult in general. However, the structure of the cost function enables us to derive some useful results that characterize the optimal solution in general, and provide necessary and sufficient conditions for optimality in some specific cases.

#### 2.3.1 Block Policy Optimality

We now provide our primary general result, in which we give a characterization of an optimal search policy in the multi-urn search problem. We begin with a definition.

**Definition 2.** A block policy is a valid policy \( u_B \) in which each urn is queried exhaustively prior to querying another urn. A block policy can be specified as a sequence of urns \( u_B = (v^1, v^2, \ldots, v^{|\mathcal{V}|}), v^i \in \mathcal{V} \), implying

\[ u(t) = v^i, \sum_{j=1}^{i-1} N_j \leq t < \sum_{j=1}^{i} N_j. \]

This definition can be used to characterize the optimal policy, which we now formally state.

**Theorem 2 (Block Policy Optimality).** Given a multi-urn search problem in which the objective is to minimize the expected number of searches required to find a red marble, an optimal search policy exists that is a block policy.

The proof of Theorem 2 is in Section 2.7, at the end of this chapter.

This result says that an optimal policy for the multi-urn search problem can be characterized by a sequence of urns. Once this is specified, one then simply searches each urn until it is out of balls or a red ball is found. The surprising part of this result is that this block policy optimality holds for arbitrary correlations in the a priori connection probabilities. For instance, there can be a negative correlation between two urns, where if the red ball is more likely to be in one urn, it is less likely to be in another. In this case one may intuitively expect that after querying an urn many times and
not finding a red ball, at some point it might be advantageous to search another urn which has a negative correlation with the queried urn. However, our result says that it is optimal to continue querying the current urn until it is exhausted.

While Theorem 2 allows for optimal policies that are not block policies, constructing such a case requires initial conditions that include probabilities that are zero. If \( \varphi_{\{i,j\}} > 0 \) for all pairs \( \{i,j\} \subset V \) (as in the case of independent urns), then only block policies can be optimal policies. This result follows from the proof of Theorem 2 (found in the Section 2.7): observe that this condition implies that function \( h(t) \) in equation (2.13) is strictly increasing in \( t \), creating a contradiction in equation (2.14).

We have shown that for multi-urn search problems, a block policy is optimal, but we have not yet specified what the block policy is. In general it can be difficult under arbitrary correlation structures to find the optimal policy. However, under certain assumptions on the urn probability model, explicit necessary and sufficient optimality conditions can be found. We examine these conditions next.

### 2.3.2 Independent Urns

We now consider the special case in which the red marbles are assumed to be independently present in each of the urns, so that the presence of a red marble in any urn (or group of urns) does not affect the probability of a red marble being present in any other urn (or group of urns). This probabilistic independence can be formalized mathematically.

**Definition 3.** An independent multi-urn search problem is a multi-urn search problem in which, for any subset of urns, \( U \subseteq V \),

\[
\varphi_U = \prod_{i \in U} \varphi_i.
\]

Intuitively, this independence property should be maintained throughout the search process for any search policy, as we now show explicitly.

**Theorem 3** (Independent Urn Probabilities). Given an independent multi-urn search problem, then for any policy \( u \), at any stage \( t \), the independence property is maintained so that

\[
P\left( \bigcap_{i \in U} A_i \bigg| x(t) \right) = \prod_{i \in U} P(A_i| x(t)).
\]

The proof of Theorem 3 is in Section 2.8, at the end of this chapter.
It follows from the result in Theorem 3 that the probability of finding a red marble in stage \( t \) is only a function of the initial conditions and number of times \( u(t) \) has been queried in the past. The number of marbles that have previously been drawn from other urns \( i \neq u(t) \) do not affect \( \mathbb{P}(w(x(t), u(t)) = 1) \).

Because of the independence of the urn probabilities, we are able to obtain closed form expressions for the probability of finding a red ball and the expected cost of a block policy, which are stated in the following results.

**Corollary 3.** Given an independent multi-urn search problem, the probability distribution of \( w(x(t), u(t)) \) at any stage \( t \) is

\[
\mathbb{P}(w(x(t), u(t)) = 0) = \frac{N_{u(t)} - x_{u(t)}(t + 1) \varphi_{u(t)}}{N_{u(t)} - x_{u(t)}(t) \varphi_{u(t)}}.
\]

**Corollary 4.** Given an independent multi-urn search problem and a block policy \( u_B = (v^1, v^2, \ldots, v^{|\mathcal{V}|}) \), \( v^i \in \mathcal{V} \), such that \( \tau(i) = \sum_{j=1}^{i-1} N_{v^j} \) is the first stage in which urn \( v^i \) is queried. Then,

\[
\prod_{t=\tau(i)}^{\tau(i)+N_i-1} \mathbb{P}(w(x(t), u(t)) = 0) = (1 - \varphi_{v^i}),
\]

the contribution of urn \( v^i \) to the total expected cost is

\[
\sum_{k=\tau(i)}^{\tau(i)+N_i-1} \prod_{t=0}^{k} \mathbb{P}(w(x(t), u(t)) = 0) = \left( N_{v^i} - \frac{(N_{v^i} + 1) \varphi_{v^i}}{2} \right) \prod_{j=1}^{i-1} (1 - \varphi_{v^j}),
\]

and the total expected cost is

\[
\mathbb{E}[C] = \sum_{i=1}^{|\mathcal{V}|} \left( N_{v^i} - \frac{(N_{v^i} + 1) \varphi_{v^i}}{2} \right) \prod_{j=1}^{i-1} (1 - \varphi_{v^j}).
\]

Independence implies that knowing the composition of marbles in urn \( i \) does not provide any additional information on the compositions of marbles in any of the other urns. Drawing a marble from urn \( u(t) \) in stage \( t \) still results in an update to this urn’s probability in stage \( t+1 \), but all other urn probabilities remain stationary in this state transition. This property enables us to characterize the optimal policy in the case of independent urns.

**Theorem 4** (Independent Urns Optimality). Given an independent multi-urn search problem, a
block policy

\[ u_B = (v^1, v^2, \ldots, v^{|\mathcal{V}|}) \]

is optimal if and only if

\[
N_{v^i} \left( \frac{2 - \varphi_{v^i}}{\varphi_{v^i}} \right) \leq N_{v^{i+1}} \left( \frac{2 - \varphi_{v^{i+1}}}{\varphi_{v^{i+1}}} \right), \quad i = 1, 2, \ldots, |\mathcal{V}| - 1. \tag{2.7}
\]

The proof of Theorem 4 is in Section 2.9, at the end of this chapter.

We note that this policy is not greedy, i.e., it does not maximize the probability of finding the red marble at each stage. Rather, the optimality condition in equation (2.7) balances the probability of immediately drawing a red marble with the probability of finding a red marble in successive draws from the same urn.

To gain intuition, consider a two-urn example in which each urn has the same probability of containing a red marble \((\varphi_1 = \varphi_2)\), but urn 1 has fewer marbles \((N_1 < N_2)\). In this case the optimality condition in equation (2.7) would have us initially draw marbles from urn 1, which has the same probability of giving us the red marble as urn 2 but requires fewer draws.

Alternatively, consider the case in which the two urns have the same number of marbles but \(\varphi_1 < \varphi_2\). In this case, the optimal policy according to equation (2.7) would have us draw from urn 2 first, which is more likely than urn 1 to give us a red marble in the same number of draws.

In order to more clearly distinguish between a greedy policy and an optimal one, we provide one more example. Consider the following independent multi-urn search problem with two urns. Urn 1 contains \(N_1 = 1\) marble and has probability \(\varphi_1 = 9/16\) of containing a red marble. Urn 2 contains \(N_2 = 2\) marbles and has probability \(\varphi_2 = 1\) of containing a red marble. This problem admits two block policies: \(u_B = (2, 1)\) and \(\tilde{u}_B = (1, 2)\).

Policy \(\tilde{u}_B\) is a greedy policy; urn 1, which has the highest immediate probability of producing a red marble \((9/16)\), is queried before urn 2. The expected number of blue marbles drawn using policy \(\tilde{u}_B\) is

\[
\mathbb{E}[\tilde{C}] = \frac{7}{16} + \left( \frac{7}{16} \right) \left( \frac{1}{2} \right) = \frac{21}{32}.
\]

Alternatively, if we follow policy \(u_B\) and draw from urn 2 first, then expected cost is

\[
\mathbb{E}[C] = \frac{1}{2},
\]

which is optimal. By accepting a slightly lower probability in the first draw, this policy guarantees
that the red marble is found in at most two draws. The optimality condition in Theorem 4, equation (2.7) provides the best balance between the immediate and long-term benefits of each query.

2.3.3 One Marble

We now turn our attention to another special case of the multi-urn search problem in which we limit the total number of red marbles among all of the urns to a single marble. In our network search scenario, this constraint would follow from assuming that the target user is connected to at most one of the known accounts on Mary’s list. This might not be a valid assumption for Mary to make, but it might apply to other search scenarios both in and out of the network context.

For example, suppose law enforcement investigators have evidence that a suspect made a single phone call from an unknown phone number during a certain period. Having obtained phone records from all likely recipients, they must efficiently search for the phone call of interest within the records of these likely recipients.

For a non-network example, suppose a hotel custodian, after servicing all of the hotel rooms, realizes he left his car keys in one of the rooms. The hotel might consist of several wings, each with different numbers of rooms, and the custodian might feel the loss was more probable in certain wings. The custodian wants to search the rooms efficiently for his keys, in order to find them before new customers begin to arrive.

This one-marble constraint imposes the strongest negative correlations between the urns: if the red marble is in urn $i$ then it cannot be in $j$, i.e.,

$$P(A_j|A_i) = 0, \quad i \neq j.$$  

Another way to characterize this constraint is to state that the events $A_1, A_2, \ldots, A_{|\mathcal{V}|}$ are disjoint. We now formalize this notion in a definition.

**Definition 4.** A single marble multi-urn search problem is a multi-urn search problem for which

$$\varphi_U = 0 \quad \forall \ U \subseteq \mathcal{V} \text{ such that } |U| > 1.$$  

We now analyze of the single marble search problem. First we observe that

$$\sum_{i \in \mathcal{V}} \varphi_i \leq 1.$$
We allow for the possibility that this sum is strictly less than one, implying there is a chance that none of the urns contain the red marble. If the sum is equal to one, then the assumption is that exactly one of the urns contains one red marble. Theorem 5 specifies the single marble urn probabilities for an arbitrary state \( x(t) \).

**Theorem 5** (Single Marble Urn Probabilities). Given a single marble multi-urn search problem and a search policy \( u \). Then, the probability that a red marble is in urn \( i \) given state \( x(t) \), and given no red marble has been found in the first \( t \) queries, is

\[
P(A_i \mid x(t)) = \frac{\left(1 - \frac{x_i(t)}{N_i}\right) \varphi_i}{1 - \sum_{j \in \mathcal{V}} \frac{x_j(t) \varphi_j}{N_j}}.
\]

The probability that the red marble is in all of the urns in any subset \( U \subset \mathcal{V} \), where \(|U| > 1\) is

\[
P \left( \bigcap_{i \in U} A_i \mid x(t) \right) = 0.
\]

**Proof.** Proof. This result follows immediately from the definition of a single marble multi-urn search problem and Theorem 1.

Because the red ball can only be in one urn, all joint probabilities are zero. This greatly simplifies our analysis and allows us to obtain closed form expressions for the probability of finding a red ball and the expected cost of a block policy, which are stated in the following results.

**Corollary 5.** Given a single marble multi-urn search problem and a valid search policy \( u \), the probability distribution of \( w(x(t), u(t)) \), conditioned on not having found a red marble in a previous stages, is

\[
P(w(x(t), u(t)) = k) = \begin{cases} 
1 - \sum_{j \in \mathcal{V}} \left( \frac{x_j(t+1) \varphi_j}{N_j} \right), & k = 0 \\
1 - \sum_{j \in \mathcal{V}} \left( \frac{x_j(t) \varphi_j}{N_j} \right), & k = 1
\end{cases}
\]

**Corollary 6.** Given a single marble multi-urn problem and a block policy \( u_B = (v^1, v^2, \ldots, v^{\lvert \mathcal{V} \rvert}) \), \( v^i \in \mathcal{V} \), such that \( \tau(i) = \sum_{j=1}^{i-1} N_{v^j} \) is the first stage in which urn \( v^i \) is queried. Then, the probability of
not finding the red marble before reaching stage \( \tau(i) \) is

\[
\tau(i) \prod_{t=0}^{\tau(i)} P(w(x(t), u(t)) = 0) = 1 - \sum_{j=1}^{i-1} \varphi_{ij},
\]

the contribution of urn \( v^i \) to the total expected cost is

\[
\sum_{k=\tau(i)}^{\tau(i)+N_i-1} \prod_{t=0}^{k} P(w(x(t), u(t)) = 0) = \left( N_i - \frac{(N_i + 1) \varphi_i}{2} - N_i \sum_{j=1}^{i-1} \varphi_j \right),
\]

and the total expected cost is

\[
E[C] = \sum_{i=1}^{\vert \mathcal{V} \vert} \left( N_{v^i} - \frac{(N_{v^i} + 1) \varphi_{v^i}}{2} \right) - \sum_{i=1}^{\vert \mathcal{V} \vert-1} \sum_{j=i+1}^{\vert \mathcal{V} \vert} N_{v^j} \varphi_{v^i}.
\]

Theorem 6 characterizes the optimal solution in the single marble case.

**Theorem 6 (Single Marble Optimality).** Given a single marble multi-urn search problem, a block policy

\[ u_B = (v^1, v^2, \ldots, v^{\vert \mathcal{V} \vert}) \]

is an optimal policy if and only if

\[
\frac{\varphi_{v^i}}{N_{v^i}} \geq \frac{\varphi_{v^{i+1}}}{N_{v^{i+1}}}, \quad i = 1, 2, \ldots, \vert \mathcal{V} \vert - 1. \tag{2.8}
\]

The proof of Theorem 6 is in Section 2.10, at the end of this chapter.

The optimality condition given in equation (2.8) leads to a greedy policy in which, at each stage, the marble that is drawn is the one that is most likely to be red. At stage \( t = 0 \) this is certainly true, as the probability of drawing a red marble from any urn \( i \) in the first draw is \( \varphi_i/N_i \), which is exactly what the optimality condition optimizes. In the next section we show that this condition is maintained through state transitions in an optimal policy. If drawing a marble from urn \( i \) has the highest probability of producing a red marble in stage \( t \), then (assuming the urn has at least one marble remaining) drawing another marble from the same urn maximizes the probability of finding a red marble in stage \( t + 1 \).
2.3.4 Monotonicity Properties

We now provide a few monotonicity properties that give additional insight into the dynamics of multi-urn search problems, as well as block policy optimality.

**Theorem 7** (Monotonicity). *Given a multi-urn search problem and a search policy \( u \), the following inequalities hold:*

1. *For any subset of urns \( U \subseteq V \) such that \( u(t) \in U \),*
   \[
   \mathbb{P} \left( \bigcap_{i \in U} A_i \mid x(t) \right) \geq \mathbb{P} \left( \bigcap_{i \in U} A_i \mid x(t + 1) \right),
   \]
   with equality holding only in cases in which \( \mathbb{P} (A_{u(t)} \mid x(t)) = 1 \) or \( \mathbb{P} (\bigcap_{i \in U} A_i \mid x(t)) = 0 \).

2. *For any stage \( t \) for which urn \( u(t) \) has more than one marble remaining, i.e., \( N_{u(t)} - x_{u(t)}(t) > 1 \),*
   \[
   \mathbb{P}(w(x(t), u(t)) = 1) \leq \mathbb{P}(w(x(t + 1), u(t)) = 1),
   \]
   with equality holding only when \( \mathbb{P}(w(x(t), u(t)) = 1) = 0 \).

The proof of Theorem 7 is in Section 2.11, at the end of this chapter.

These monotonicity properties provide intuition into why optimal block policies exist. Suppose at stage \( t \) a blue marble is drawn from urn \( u(t) \). At stage \( t + 1 \), the probability of urn \( u(t) \) containing a red marble has decreased as a result of this new information. However, the probability that the next marble drawn from urn \( u(t) \) is red has *increased* from the previous stage. If drawing from urn \( u(t) \) in stage \( t \) had a high probability of returning a red marble, drawing another marble from \( u(t) \) in stage \( t + 1 \) has an even higher probability of producing a red marble.

In the case of independent urns, this property provides justification for using a block policy. Suppose urn \( u(t) \) has multiple marbles in it and is optimal at stage \( t \), and a blue marble is drawn from this urn. At stage \( t + 1 \) the probability of drawing a red marble from urn \( u(t) \) has increased, while all other urn and marble probabilities have remained unchanged from the previous stage \( t \). It follows that it would continue to be optimal to draw from urn \( u(t) \).

If we allow for correlations among the urns, however, drawing a blue marble from urn \( u(t) \) might also increase the probability of drawing a red marble from other urns in the following stage. In the single marble multi-urn search problem, drawing a blue marble from urn \( u(t) \) in stage \( t \) increases the probability of finding a red marble in each of the other urns in stage \( t + 1 \). We now provide our
final theoretical result, which states that the rate of probability growth in a queried urn is always at least as large as the rate of probability growth in any other urn.

**Theorem 8** (Marble Probability Bound). *Given a multi-urn search problem and a search policy $u$, the following inequality holds:*

$$\frac{\mathbb{P}(w(x(t+1), u(t)) = 1)}{\mathbb{P}(w(x(t), u(t)) = 1)} \geq \frac{\mathbb{P}(w(x(t+1), i) = 1)}{\mathbb{P}(w(x(t), i) = 1)}.$$  

The proof of Theorem 8 is in Section 2.12, at the end of this chapter.

Theorem 8 provides much intuition about why optimal block policies always exist in multi-urn search problems. It also shows that a purely greedy strategy, in which the probability of immediately drawing a red marble is maximized at each stage, will always produce a (possibly suboptimal) block policy. Finally, it provides some insight into why the optimality conditions for the single marble multi-urn search problem given in equation (2.8) result in a greedy policy. Equation (2.8) specifies that the first marble drawn is the one that maximizes the probability of immediately finding the target. It follows from Theorem 8 that subsequent draws from the same urn will continue to maximize this probability.

### 2.3.5 Urn Correlation Dynamics

In the preceding section we examined how urn probabilities and the probability of drawing a red marble from each urn evolved as a function of the state of the system. In this section we show by example how, in general, the correlations among the urns can evolve in ways that we find to be counterintuitive.

We say that two urns $i$ and $j$ are *positively correlated* at stage $t$ if they have positive covariance, i.e.,

$$\mathbb{P}(A_i \cap A_j | x(t)) > \mathbb{P}(A_i | x(t)) \mathbb{P}(A_j | x(t)).$$

Likewise, urns $i$ and $j$ are *negatively correlated* if their covariance is negative,

$$\mathbb{P}(A_i \cap A_j | x(t)) < \mathbb{P}(A_i | x(t)) \mathbb{P}(A_j | x(t)).$$

In Theorem 3 we showed the somewhat intuitive result that independence among the urn probabilities is preserved through state transitions. In general, correlations can change through Bayesian updates each time a blue marble is drawn. These changes can include changes in sign.
We now provide an example scenario in which all correlations are positive in the initial conditions, but after a blue marble is drawn some correlations become negative. Suppose we have a multi-urn search problem consisting of three urns. Each urn $i$ has $N_i = 1$ marble and initial probability $\varphi_i = \frac{1}{2}$ of containing a red marble. Furthermore,

$$\varphi_{\{1,2\}} = \varphi_{\{1,3\}} = \varphi_{\{2,3\}} = \varphi_{\{1,2,3\}} = \frac{1}{3}.$$ 

A quick calculation confirms that this is a valid probability model, and that a red marble exists in at least one of the three urns with probability $\frac{5}{6}$. We also verify that all correlations are positive,

$$\varphi_{\{1,2\}} > \varphi_1 \varphi_2$$
$$\varphi_{\{1,3\}} > \varphi_1 \varphi_3$$
$$\varphi_{\{2,3\}} > \varphi_2 \varphi_3.$$ 

Also, we have a more general positive correlation property,

$$\varphi_{\{1,2,3\}} > \varphi_1 \varphi_2 \varphi_3.$$ 

Figure 2-2 depicts this probability law in a Venn diagram. Note that there is no probability of exactly two urns containing red marbles. A red marble is present in zero, one, or three urns almost surely.

Now suppose at stage $t = 0$, a blue marble is drawn from urn 1. From Theorem 1, the stage 1
urn probabilities are

\[ P(A_1 \mid x(1)) = 0 \]
\[ P(A_2 \mid x(1)) = \frac{1}{3} \]
\[ P(A_3 \mid x(1)) = \frac{1}{3} \]
\[ P(A_1 \cap A_2 \mid x(1)) = 0 \]
\[ P(A_1 \cap A_3 \mid x(1)) = 0 \]
\[ P(A_2 \cap A_3 \mid x(1)) = 0 \]
\[ P(A_1 \cap A_2 \cap A_3 \mid x(1)) = 0. \]

By eliminating the possibility that each urn contained a red marble, the only outcomes that have positive probability in stage 1 are single-marble outcomes. The correlation between urns 2 and 3, which was positive in the initial conditions, has become negative in stage 1:

\[ P(A_2 \cap A_3 \mid x(1)) = 0 < \frac{1}{9} = P(A_2 \mid x(1)) P(A_3 \mid x(1)). \]

One could similarly produce examples in which correlations that were originally negative become positive after drawing blue marbles. In the single-marble and independent urn cases, for which we provided characterizations of the optimal policies in the preceding sections, correlations among the urns exhibited some stationarity with respect to stage. In general, the nature of correlations among the urns can change substantially as blue marbles are drawn and probabilities are updated. This characteristic presents a challenge to finding characterizations of the optimal policy in general.

### 2.4 Summary of Network Search Contributions

We have presented a multi-urn search problem as a model for searching for a specific vertex in a network. Using this model, we have shown that there is always an optimal block policy in searches that meet the multi-urn search problem assumptions, irrespective of correlations in the probability model. We have also provided necessary and sufficient conditions for block policy optimality in two specific cases: independent urns and the single red marble scenario. Finally, we gave a few properties of the dynamics of the multi-urn search problem and commented on the challenges of finding more general optimality conditions.
There are additional generalizations and extensions that we have not considered here, but which might also have interesting applications in modeling search. One such generalization is removing the constraint that an urn can have at most one red marble. This generalization might be an appropriate model for a network vertex search problem in which the network structure allowed for multiple edges between a pair of nodes. Allowing for multiple red marbles in a single urn substantially changes the dynamics of the multi-urn search problem.

Another area of further inquiry could involve examination of the performance of different policies under various urn probability models. We have shown, for example, that a purely greedy policy is not optimal in the case of independent urns. However, the counter-example suggests that there could be some lower bounds on greedy policy performance, which might depend on the total number of urns and total number of marbles. At the very least, there appears to be limits on how suboptimal we can make a greedy policy when constructing a two-urn data set. Development in this direction could build on the results presented by Chen et al. [31].

We have assumed throughout our analysis that the target of the search would be easily identifiable to the searcher. In our urn model this assumption translated to clear color distinction, so we assume we know immediately whether a drawn marble is red or blue. However, we could relax this assumption in several ways. We could, for example, characterize each marble with a feature set and develop a probability model that gives us the probability that a marble is a red marble, given its set of features. Depending on the context of the search, this type of incomplete information model could evolve into a stopping problem in which the objective is to determine the best time to stop drawing new marbles in search of a red one. Multi-arm bandit models and sequential mutual information maximization [31] might offer useful approaches in this scenario.

A slight variation from this imperfect information approach would be to represent some marbles as being more “red” than others, according to some probability model. In this model, the searcher receives a reward at the end of the search that is a function of the most “red” marble found, but still has to pay a fixed cost for each draw. Like the imperfect information model, this formulation would ultimately be a stopping problem, balancing the current reward attained against the likelihood of attaining a higher reward by drawing more marbles.

We have also assumed uniform urn costs. It is plausible, however, that in some cases it might cost more to draw marbles from some urns than from others. Or, there could be a one-time access cost in order to gain the ability to draw marbles from the urn, which could represent law enforcement having to get a warrant to obtain Internet or phone records for an individual.
Our assumption that marbles are drawn in a random sequence from each urn might not be valid in some settings. The Google search engine, for example, returns the most relevant results first, so that if a user does not find what he is looking for in the first few pages of results it might make sense to try a different query rather than look through the remainder of the pages. If we changed our model so that red marbles were more likely to be drawn first in each urn, then the problem would involve deciding when to stop querying an urn and switch to one that might be more promising.

The technical proofs of the statements in this chapter are provided in the next section. In Chapter 3 of this thesis, we apply the theoretical results developed in this chapter to the problem of finding new Twitter accounts belong to known extremists after their previous accounts have been suspended.

2.5 Proofs of Statements

In this section we provide the technical proofs for the theorems provided in this chapter.

2.6 Proof of Theorem 1

Theorem. In a multi-urn search problem over a set of urns \( \mathcal{V} \), suppose a red marble is not found in the first \( t \) queries when executing a valid policy \( u = (u(0), \ldots, u(t - 1)) \). Then, for any subset of urns \( U \subseteq \mathcal{V} \), the probability of a red marble being in all of the urns in \( U \) at stage \( t \) is given by:

\[
\mathbb{P} \left( \bigcap_{i \in U} A_i \bigg| x(t) \right) = \left[ \prod_{i \in U} \left( 1 - \frac{x_i(t)}{N_i} \right) \right] \sum_{S \subseteq \mathcal{V} : S \supseteq U} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j} \sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j}.
\]

Proof of Theorem 1. Substituting the initial condition, \( x(0) = 0 \), into the result returns the prior

\[
\mathbb{P} \left( \bigcap_{i \in U} A_i \bigg| x(0) \right) = \varphi_U := \mathbb{P} \left( \bigcap_{i \in U} A_i \right).
\]

The proof proceeds by induction. First note that in order to reach state \( x(t + 1) \) from stage \( t \), a blue marble must have been drawn from urn \( u(t) \) from state \( x(t) \). We use the law of total
probability to decompose this event and form a recursion:

\[
P\left( \bigcap_{i \in U} A_i \bigg| x(t+1) \right) = \frac{P \left( (\bigcap_{i \in U} A_i) \cap w(x(t), u(t)) = 0 | x(t) \right)}{P(w(x(t), u(t)) = 0)}
\]

\[
= \left( 1 - \frac{1}{N_{u(t)} - x_{u(t)}(t)} \right) P \left( \bigcap_{i \in U \cup \{u(t)\}} A_i \bigg| x(t) \right) + \frac{P \left( A_{u(t)}^c \cap (\bigcap_{i \in U} A_i) | x(t) \right)}{1 - \left( \frac{1}{N_{u(t)} - x_{u(t)}(t)} \right) P(A_{u(t)}|x(t))}
\]

\[
= \frac{P \left( \bigcap_{i \in U} A_i \bigg| x(t) \right) - \left( \frac{1}{N_{u(t)} - x_{u(t)}(t)} \right) P \left( \bigcap_{i \in U \cup \{u(t)\}} A_i \bigg| x(t) \right)}{1 - \left( \frac{1}{N_{u(t)} - x_{u(t)}(t)} \right) P(A_{u(t)}|x(t))}
\] (2.9)

The result in Theorem 1 forms our induction hypothesis. We use this result to form the three probabilities in the recursion given in equation (2.9).

\[
P \left( \bigcap_{i \in U} A_i \bigg| x(t) \right) = \left[ \prod_{i \in U} \left( 1 - \frac{x_i(t)}{N_i} \right) \right] \sum_{S \subseteq \mathcal{V}: S \supseteq U} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j}
\]

\[
P \left( A_{u(t)}|x(t) \right) = \left( 1 - \frac{x_{u(t)}(t)}{N_{u(t)}} \right) \sum_{S \subseteq \mathcal{V}: u(t) \in S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{u(t)\}} \frac{x_j(t)}{N_j}
\]

\[
P \left( \bigcap_{i \in U \cup \{u(t)\}} A_i \bigg| x(t) \right) = \left\{ \begin{array}{ll}
\left[ \prod_{i \in U} \left( 1 - \frac{x_i(t)}{N_i} \right) \right] \sum_{S \subseteq \mathcal{V}: S \supseteq U} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j} & \text{if } u(t) \in U \\
\left[ \prod_{i \in (U \cup \{u(t)\})} \left( 1 - \frac{x_i(t)}{N_i} \right) \right] \sum_{S \subseteq \mathcal{V}: (U \cup \{u(t)\}) \subseteq S} (-1)^{|S| - |U| - 1} \varphi_S \prod_{j \in S \setminus (U \cup \{u(t)\})} \frac{x_j(t)}{N_j} & \text{if } u(t) \notin U.
\end{array} \right.
\]

As we see from these probabilities, we have two cases to consider:

1. \( u(t) \in U \).

2. \( u(t) \notin U \).

We now look at each of these cases individually.

**Case 1:** \( u(t) \in U \). We begin by substituting the probabilities formed using the induction
observe that this is the desired result for stage $hypothesis into the recursion in equation (2.9).

$$
P \left( \bigcap_{i \in U} A_i \bigg| x(t+1) \right) = \frac{\mathbb{P} \left( \bigcap_{i \in U \cup \{u(t)\}} A_i \big| x(t) \right) - \left( \frac{1}{N_{u(t)}-x_{u(t)}(t)} \right) \mathbb{P} \left( \bigcap_{i \in U \cup \{u(t)\}} A_i \big| x(t) \right)}{1 - \left( \frac{1}{N_{u(t)}-x_{u(t)}(t)} \right) \mathbb{P}(A_{u(t)} \big| x(t))} = \left( 1 - \frac{1}{N_{u(t)}-x_{u(t)}(t)} \right) \left( \prod_{i \in U \cup \{u(t)\}} \left( 1 - \frac{x_i(t)}{N_i} \right) \right) \sum_{S \subseteq V: \, S \supseteq U} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j}
$$

We proceed by separating the summations into terms corresponding to sets containing $u(t)$ and those that do not contain $u(t)$. We can then factor out the terms corresponding to urn $u(t)$ and make the following substitutions:

$$x_i(t) = \begin{cases} 
x_i(t+1) & i \neq u(t) 
x_i(t+1) - 1 & i = u(t). 
\end{cases}$$

Continuing the simplification from above,

$$\begin{align*}
&= \left( \frac{N_{u(t)} - x_{u(t)}(t+1)}{N_{u(t)}} \right) \left[ \prod_{i \in U \setminus \{u(t)\}} \left( 1 - \frac{x_i(t+1)}{N_i} \right) \right] \sum_{S \subseteq V: \, S \supseteq U} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t+1)}{N_j} \\
&= \left[ \prod_{i \in U} \left( 1 - \frac{x_i(t+1)}{N_i} \right) \right] \sum_{S \subseteq V: \, S \supseteq U} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S} \frac{x_j(t+1)}{N_j} \\
&= \left[ \prod_{i \in U} \left( 1 - \frac{x_i(t+1)}{N_i} \right) \right] \sum_{S \subseteq V: \, S \supseteq U} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t+1)}{N_j}.
\end{align*}$$

Observe that this is the desired result for stage $t + 1$. We now provide the induction step for the case in which $u(t) \notin U$. 

63
Case 2: \( u(t) \notin U \) follows a similar set of steps. We begin by substituting the probabilities formed using the induction hypothesis into the recursion in equation (2.9).

\[
\frac{\mathbb{P}(\bigcap_{i \in U} A_i | x(t)) - \left(\frac{1}{N_{u(t)} - x_{u(t)}(t)}\right) \mathbb{P}(\bigcap_{i \in U \cup u(t)} A_i | x(t))}{1 - \left(\frac{1}{N_{u(t)} - x_{u(t)}(t)}\right) \mathbb{P}(A_{u(t)} | x(t))} \\
= \left[ \left(\prod_{i \in U} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \sum_{\{S \subseteq V: S \supseteq U\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j} \right] \\
- \left(\frac{1}{N_{u(t)} - x_{u(t)}(t)}\right) \left(\prod_{i \in U \cup u(t)} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \sum_{\{S \subseteq V: S \supseteq U \cup u(t)\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus (U \cup u(t))} \frac{x_j(t)}{N_j} \\
\times \left[ 1 - \left(\frac{1}{N_{u(t)} - x_{u(t)}(t)}\right) \left(\prod_{i \in U \cup u(t)} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \sum_{\{S \subseteq V: u(t) \in S\}} (-1)^{|S| - |U| - 1} \varphi_S \prod_{j \in S \setminus \{u(t)\}} \frac{x_j(t)}{N_j} \right]^{-1}
\]

The denominators in the above expression reduce in exactly the same way as in the previous case in which \( u(t) \in U \). In fact, the denominators in the induction hypothesis and in equation (2.9) do not depend on whether \( u(t) \in U \). Because we have already shown the steps for reducing this denominator to the desired form, we omit these steps and only show the induction on the numerators for this case:

\[
\left[ \left(\prod_{i \in U} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \sum_{\{S \subseteq V: S \supseteq U\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j} \right] \\
- \left(\frac{1}{N_{u(t)} - x_{u(t)}(t)}\right) \left(\prod_{i \in U \cup u(t)} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \sum_{\{S \subseteq V: S \supseteq U \cup u(t)\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus (U \cup u(t))} \frac{x_j(t)}{N_j} \\
= \left(\prod_{i \in U} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \left[ \sum_{\{S \subseteq V: S \supseteq U\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j} \right] \\
+ \left(1 \frac{1}{N_{u(t)} - x_{u(t)}(t)}\right) \left(\prod_{i \in U \cup u(t)} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \sum_{\{S \subseteq V: S \supseteq U \cup u(t)\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus (U \cup u(t))} \frac{x_j(t)}{N_j} \\
= \left(\prod_{i \in U} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \left[ \sum_{\{S \subseteq V \setminus \{u(t)\}: S \supseteq U\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j} \right] \\
+ \left(\frac{1}{N_{u(t)} - x_{u(t)}(t)}\right) \left(\prod_{i \in U \cup u(t)} \left(1 - \frac{x_i(t)}{N_i}\right) \right) \sum_{\{S \subseteq V: S \supseteq U \cup u(t)\}} (-1)^{|S| - |U|} \varphi_S \prod_{j \in S \setminus (U \cup u(t))} \frac{x_j(t)}{N_j} \\
\right]
\]
We again make the substitution:

\[
x_i(t) = \begin{cases} 
  x_i(t+1) & i \neq u(t) \\
  x_i(t+1) - 1 & i = u(t),
\end{cases}
\]

and continue from above:

\[
= \left( \prod_{i \in U} \left( 1 - \frac{x_i(t+1)}{N_i} \right) \right) \left[ \sum_{S : S \subseteq \mathcal{V} \setminus \{u(t)\} : S \supseteq U} (-1)^{|S|-|U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t+1)}{N_j} \right] \\
+ \left( \frac{x_{u(t)}(t+1)}{N_{u(t)}} \right) \sum_{S : S \subseteq \mathcal{V} : S \supsetneq U \cup \{u(t)\}} (-1)^{|S|-|U|} \varphi_S \prod_{j \in S \setminus (U \cup \{u(t)\})} \frac{x_j(t+1)}{N_j}
\]

\[
= \left( \prod_{i \in U} \left( 1 - \frac{x_i(t+1)}{N_i} \right) \right) \sum_{S : S \subseteq \mathcal{V} : S \supseteq U} (-1)^{|S|-|U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t+1)}{N_j}.
\]

This final expression is the numerator in Theorem 1 for the stage \( t+1 \) urn probabilities. \( \square \)

### 2.7 Proof of Theorem 2

**Theorem.** Given a multi-urn search problem in which the objective is to minimize the expected number of searches required to find a red marble, an optimal search policy exists that is a block policy.

#### 2.7.1 Preparatory Material

**Lemma 1.** Given a multi-urn search problem on a set of urns \( \mathcal{V} \), suppose a policy \( u \) is executed to stage \( t \) irrespective of whether a red marble is found at any stage. Let \( B_i \) be the event that a red marble has been drawn from urn \( i \in \mathcal{V} \) in this experiment. Then, for any subset \( U \subseteq \mathcal{V} \), the probability of having drawn a red marble from each of the urns in \( U \) and none of the other urns is

\[
\Pr \left( \left( \bigcap_{i \in U} B_i \right) \cap \left( \bigcap_{j \in \mathcal{V} \setminus U} B_j \right) \right) = \sum_{S : S \subseteq \mathcal{V} : S \supseteq U} (-1)^{|S|-|U|} \varphi_S \prod_{i \in S} \frac{x_i(t)}{N_i} \geq 0.
\]
Proof. Proof of Lemma 1. This result is comes from the principle of inclusion-exclusion, and follows from Equation 2.1. From basic set operations and DeMorgan’s Law, we can write

\[
\left( \bigcap_{i \in U} B_i \right) = \left( \bigcap_{i \in U} B_i \right) \cap \left( \bigcap_{j \in V \setminus U} B_j^c \right) \cup \left( \bigcap_{i \in U} B_i \right) \cap \left( \bigcup_{j \in V \setminus U} B_j \right),
\]

which is a union of disjoint sets. Therefore,

\[
P \left( \left( \bigcap_{i \in U} B_i \right) \cap \left( \bigcap_{j \in V \setminus U} B_j^c \right) \right) = P \left( \bigcap_{i \in U} B_i \right) - P \left( \left( \bigcap_{i \in U} B_i \right) \cap \left( \bigcup_{j \in V \setminus U} B_j \right) \right). \tag{2.10}
\]

The probability a red marble is drawn from all of the urns in \( U \) in this experiment can be found using the multiplication rule:

\[
P \left( \bigcap_{i \in U} B_i \right) = \varphi_U \prod_{i \in U} \frac{x_i(t)}{N_i}. \tag{2.11}
\]

Recall that \( \varphi_U \) is the probability of all of the urns in set \( U \) containing a red marble, and \( \frac{x_i(t)}{N_i} \) is simply the fraction of marbles removed from urn \( i \) at stage \( t \). The product in this expression implies conditional independence: given all of the urns in \( U \) contain a red marble, the probability that a red marble is drawn from each of them by stage \( t \) is the product of the individual probabilities. This conditional independence follows implicitly from our search assumptions. The order of marble draws from each urn is random, and does not depend on the order of marble draws from any other urn.

We now examine \( P \left( \left( \bigcap_{i \in U} B_i \right) \cap \left( \bigcup_{j \in V \setminus U} B_j \right) \right) \). We first note that

\[
\left( \bigcap_{i \in U} B_i \right) \cap \left( \bigcup_{j \in V \setminus U} B_j \right) = \bigcup_{j \in V \setminus U} \left( \bigcap_{i \in U} B_i \right) \cap B_j.
\]

66
Using the principle of inclusion exclusion, we can find the probability of this union,

\[
P\left(\bigcup_{j \in \mathcal{V}\setminus U}\left(\bigcap_{i \in U}B_i \cap B_j\right)\right) = \sum_{j \in \mathcal{V}\setminus U} P\left(\bigcap_{i \in U}B_i \cap B_j\right) - \sum_{j \in \mathcal{V}\setminus U}\left[\sum_{k \in \mathcal{V}\setminus (U \cup \{j\})} P\left(\bigcap_{i \in U}B_i \cap B_j \cap B_k\right)\right]
\]

\[
\cdots + (-1)^{|\mathcal{V}|-|U|+1} P\left(\bigcap_{i \in \mathcal{V}}B_i\right).
\]

(2.12)

Substituting expressions 2.11 and 2.12 into equation 2.10 reduces to the desired result. The principle of inclusion-exclusion and the axioms of probability ensure that this quantity is nonnegative. \(\square\)

2.7.2 Proof of Main Result

Proof. Proof of Theorem 2. Suppose we are given a multi-urn search problem on a set of urns \(\mathcal{V}\), with each urn \(i \in \mathcal{V}\) containing \(N_i\) marbles and initial target probabilities \(\varphi_U\) for all \(U \subseteq \mathcal{V}\). Suppose also that valid policy \(u = (u(0), \ldots, u(N - 1))\) is optimal, where \(u\) is not a block policy. This implies that we can find a stage \(\tau\) where

\[
\begin{align*}
    u(\tau) &= i \\
u(\tau + 1), u(\tau + 2), \ldots, u(\tau + \delta - 1) &\neq i \\
u(\tau + \delta) &= i,
\end{align*}
\]

for some urn \(i \in \mathcal{V}\), where \(\delta > 1\).

We now consider two alternative policies that move the queries of urn \(i\) into “blocks”. Policy \(\hat{u}\) executes the two queries of \(i\) in stages \(\tau\) and \(\tau + 1\), then executes the rest of the queries in the subsequence. Policy \(\tilde{u}\) executes the two queries of \(i\) after executing the other queries in the subsequence. A visual comparison of these policies is provided in Figure 2-3. Formally,

\[
\hat{u} = \hat{u}(t) = \begin{cases} 
    u(t), & t \leq \tau \text{ or } t > \tau + \delta \\
    u(t - 1), & t = \tau + 1, \tau + 2, \ldots, \tau + \delta,
\end{cases}
\]

\[
\tilde{u} = \tilde{u}(t) = \begin{cases} 
    u(t), & t < \tau \text{ or } t \geq \tau + \delta \\
    u(t + 1), & t = \tau, \tau + 1, \ldots, \tau + \delta - 1.
\end{cases}
\]

67
Let $C$ be the number of non-target queries (i.e., the cost) when using policy $u$, $\hat{C}$ be the same quantity when using policy $\hat{u}$, and $\tilde{C}$ be the same quantity when using policy $\tilde{u}$. Conditioned on not having found a red marble, it follows that the state trajectories,

$$\hat{x}_j(t) = \begin{cases} 
  x_j(t), & t \leq \tau + 1 \text{ or } t > \tau + \delta \\
  x_j(t - 1), & t = \tau + 2, \ldots, \tau + \delta, \ j \neq i \\
  x_j(t - 1) + 1, & t = \tau + 2, \ldots, \tau + \delta, \ j = i 
\end{cases}$$

$$\tilde{x}_j(t) = \begin{cases} 
  x_j(t), & t \leq \tau \text{ or } t \geq \tau + \delta \\
  x_j(t + 1), & t = \tau + 1, \ldots, \tau + \delta - 1, \ j \neq i \\
  x_j(t + 1) - 1, & t = \tau + 1, \ldots, \tau + \delta - 1, \ j = i,
\end{cases}$$

where $\hat{x}_j(t)$ and $\tilde{x}_j(t)$ are the numbers of times urn $j$ has been queried before stage $t$ under policies $\hat{u}$ and $\tilde{u}$, respectively.
Using equation (2.5), our assumptions imply that

\[
E[\hat{C}] - E[C] \geq 0
\]

\[
\sum_{t=0}^{N-1} \sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t+1)}{N_j} - \sum_{t=0}^{N-1} \sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t+1)}{N_j} \geq 0
\]

\[
\sum_{S \subseteq \mathcal{V}\setminus\{i\}} \left( \sum_{t=r+2}^{r+\delta} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j} - \sum_{t=r+2}^{r+\delta} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j} \right)
\]

\[
+ \sum_{S \subseteq \mathcal{V}\setminus\{i\}} \left( \sum_{t=r+1}^{r+\delta-1} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j} - \sum_{t=r+1}^{r+\delta-1} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j} \right)
\]

\[
+ \sum_{S \subseteq \mathcal{V}\setminus\{i\}} \left( (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(\tau+1)}{N_j} - (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(\tau+1)}{N_j} \right)
\]

\[
+ \frac{1}{N_i} \sum_{S \subseteq \mathcal{V}\setminus\{i\}} \sum_{t=r+2}^{r+\delta-1} (-1)^{|S|} \varphi_S \prod_{j \in S\setminus\{i\}} \frac{x_j(t)}{N_j} \geq 0
\]

\[
\sum_{S \subseteq \mathcal{V}\setminus\{i\}} \left( (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(\tau+\delta)}{N_j} - (-1)^{|S|} \varphi_S \prod_{j \in S\setminus\{i\}} \frac{x_j(\tau+1)}{N_j} \right)
\]

\[
+ \sum_{S \subseteq \mathcal{V}\setminus\{i\}} \left( (-1)^{|S|} \varphi_S \prod_{j \in S\setminus\{i\}} \frac{x_j(\tau+\delta)}{N_j} - (-1)^{|S|} \varphi_S \prod_{j \in S\setminus\{i\}} \frac{x_j(\tau+1)}{N_j} \right)
\]

\[- \frac{1}{N_i} \sum_{S \subseteq \mathcal{V}\setminus\{i\}} \sum_{t=r+2}^{r+\delta-1} (-1)^{|S|} \varphi_S \prod_{j \in S\setminus\{i\}} \frac{x_j(t)}{N_j} \leq \frac{1}{N_i} \sum_{S \subseteq \mathcal{V}\setminus\{i\}} \sum_{S\setminus\{i\}} (-1)^{|S|} \varphi_S \prod_{j \in S\setminus\{i\}} \frac{x_j(\tau+1)}{N_j}.
\]
Optimality of $\mathbf{u}$ likewise implies

$$
\mathbb{E}[\hat{C}] - \mathbb{E}[C] \geq 0
$$

$$
\sum_{t=0}^{N-1} \sum_{s \subseteq \mathcal{V}} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t+1)}{N_j} - \sum_{t=0}^{N-1} \sum_{s \subseteq \mathcal{V}} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t+1)}{N_j} \geq 0
$$

$$
\sum_{s \subseteq \mathcal{V}} \left( \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} - \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} \right) \geq 0
$$

$$
\sum_{s \subseteq \mathcal{V} \setminus \{i\}} \left( \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} - \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} \right) \geq 0
$$

$$
\sum_{s \subseteq \mathcal{V} \setminus \{i\}} \left( \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} \right) \geq 0
$$

$$
\sum_{s \subseteq \mathcal{V}} \left( \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} - \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} \right) \geq 0
$$

$$
\sum_{s \subseteq \mathcal{V} \setminus \{i\}} \left( \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t)}{N_j} \right) \geq 0
$$

Now let

$$
\alpha = \sum_{s \subseteq \mathcal{V} \setminus \{i\}} \left( (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t+1)}{N_j} - (-1)^{|s|} \varphi_s \prod_{j \in s} \frac{x_j(t+1)}{N_j} \right)
$$

$$
+ \sum_{s \subseteq \mathcal{V} \setminus \{i\}} \left( \sum_{j \in s} \frac{x_j(t+1)}{N_i} \right) \left( (-1)^{|s|} \varphi_s \prod_{j \in s \setminus \{i\}} \frac{x_j(t+1)}{N_j} - (-1)^{|s|} \varphi_s \prod_{j \in s \setminus \{i\}} \frac{x_j(t+1)}{N_j} \right)
$$

$$
- \frac{1}{N_i} \sum_{s \subseteq \mathcal{V} \setminus \{i\}} \sum_{t=r+1}^{t+\delta-1} (-1)^{|s|} \varphi_s \prod_{j \in s \setminus \{i\}} \frac{x_j(t)}{N_j}.
$$
which appears in both of the expected cost inequalities we have derived. We have established that

\[ \alpha \geq \left( \frac{1}{N_i} \right) \sum_{S \subseteq \mathcal{V} : i \in S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i\}} \frac{x_j(\tau + \delta)}{N_j} \]

\[ \alpha \leq \left( \frac{1}{N_i} \right) \sum_{S \subseteq \mathcal{V} : i \in S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i\}} \frac{x_j(\tau + 1)}{N_j} \]

We now show that for any \( i \in \mathcal{V} \), for any \( u(t) \in \mathcal{V} \setminus \{i\} \),

\[ h(t) = \sum_{S \subseteq \mathcal{V} : i \in S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i\}} \frac{x_j(t)}{N_j} \] \hspace{1cm} (2.13)

is a nondecreasing function that is strictly increasing when \( \varphi_{\{i, u(t)\}} > 0 \).

Observe that

\[ h(t + 1) = \sum_{S \subseteq \mathcal{V} : i \in S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i\}} \frac{x_j(t + 1)}{N_j} \]

\[ = \sum_{S \subseteq \mathcal{V} \setminus \{u(t)\} : i \in S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i\}} \frac{x_j(t)}{N_j} \]

\[ + \left( \frac{x_{u(t)}(t + 1)}{N_{u(t)}} \right) \sum_{S \subseteq \mathcal{V} : \{i, u(t)\} \subseteq S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i, u(t)\}} \frac{x_j(t)}{N_j} \]

\[ = h(t) + \left( \frac{1}{N_{u(t)}} \right) \sum_{S \subseteq \mathcal{V} : \{i, u(t)\} \subseteq S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i, u(t)\}} \frac{x_j(t)}{N_j} \]

From Lemma 1,

\[ \left( \frac{x_i(t)x_{u(t)}(t)}{N_iN_{u(t)}} \right) \sum_{S \subseteq \mathcal{V} : \{i, u(t)\} \subseteq S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i, u(t)\}} \frac{x_j(t)}{N_j} \geq 0 \]

\[ \Rightarrow \left( \frac{1}{N_{u(t)}} \right) \sum_{S \subseteq \mathcal{V} : \{i, u(t)\} \subseteq S} (-1)^{|S|} \varphi_S \prod_{j \in S \setminus \{i, u(t)\}} \frac{x_j(t)}{N_j} \geq 0. \]

This final inequality implies \( h(t + 1) \geq h(t) \). Because urn \( i \) is not queried in stages \( t = \tau + 1, \ldots, \tau + \delta \), \( h(t) \) is nondecreasing over these stages. Therefore we can write

\[ \alpha \geq h(\tau + \delta) \geq h(\tau + 1) \geq \alpha. \] \hspace{1cm} (2.14)

This expression can only be satisfied by equality, which means that for any optimal policy that is not a block policy, we can maintain optimality while successively permuting the policy so that
the urn queries are arranged into blocks.

2.8 Proof of Theorem 3

\textit{Theorem.} Given an independent multi-urn search problem, then for any policy \( u \), at any stage \( t \), the independence property is maintained so that

\[
\mathbb{P}\left( \bigcap_{i \in U} A_i \mid x(t) \right) = \prod_{i \in U} \mathbb{P}(A_i \mid x(t)).
\]

\textit{Proof.} Proof of Theorem 3. This result comes from substituting the appropriate products into the result from Theorem 1. For any urn \( i \in \mathcal{V} \),

\[
\mathbb{P}(A_i \mid x(t)) = \frac{\left(1 - \frac{x_i(t)}{N_i}\right) \sum_{S \subseteq \mathcal{V} : i \in S} (-1)^{|S|-1} \varphi_S \prod_{j \in S \setminus \{i\}} \frac{x_j(t)}{N_j}}{\sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j}}.
\]

\[
= \frac{\varphi_i \left(1 - \frac{x_i(t)}{N_i}\right) \sum_{S \subseteq \mathcal{V} \setminus \{i\}} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}}{\sum_{S \subseteq \mathcal{V} \setminus \{i\}} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}}.
\]

\[
= \frac{\varphi_i \left(1 - \frac{x_i(t)}{N_i}\right) \sum_{S \subseteq \mathcal{V} \setminus \{i\}} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}}{\sum_{S \subseteq \mathcal{V} \setminus \{i\}} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}}.
\]

\[
= \frac{\varphi_i \left(1 - \frac{x_i(t)}{N_i}\right)}{1 - \varphi_i \left( \frac{x_i(t)}{N_i} \right)}.
\]
In a similar manner we use Theorem 1 to find the urn probability for subset $U \subseteq \mathcal{V}$ at stage $t$,

$$
\mathbb{P}\left(\bigcap_{i \in U} A_i\right) = \frac{\left[\prod_{i \in U} \left(1 - \frac{x_i(t)}{N_i}\right)\right] \sum_{S \subseteq \mathcal{V}, S \supseteq U} (-1)^{|S|-|U|} \varphi_S \prod_{j \in S \setminus U} \frac{x_j(t)}{N_j}}{\sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{j \in S} \frac{x_j(t)}{N_j}}
$$

$$=
\frac{\left[\prod_{i \in U} \varphi_i \left(1 - \frac{x_i(t)}{N_i}\right)\right] \sum_{S \subseteq \mathcal{V} \setminus U} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}}{\sum_{T \subseteq U} \sum_{S \subseteq \mathcal{V} \setminus U} (-1)^{|S|+|T|} \prod_{j \in S \cup T} \varphi_j \frac{x_j(t)}{N_j}}
$$

$$=
\frac{\left[\prod_{i \in U} \varphi_i \left(1 - \frac{x_i(t)}{N_i}\right)\right] \sum_{S \subseteq \mathcal{V} \setminus U} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}}{\sum_{S \subseteq \mathcal{V} \setminus (U \cup T)} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}}
$$

$$=
\frac{\left[\prod_{i \in U} \varphi_i \left(1 - \frac{x_i(t)}{N_i}\right)\right]}{\sum_{S \subseteq \mathcal{V} \setminus U} (-1)^{|S|} \prod_{j \in S} \varphi_j \frac{x_j(t)}{N_j}} \prod_{i \in U} \mathbb{P}(A_i).
$$

In the final equality, we have used the property that for any $\beta_1, \ldots, \beta_M$,

$$\prod_{i=1}^{M} (1 - \beta_i) = \sum_{S \subseteq [M]} \prod_{j \in S} (-\beta_j).$$

We can verify this property by induction. Define $\prod_{j \in \emptyset} (-\beta_j) = 1$. Now observe

$$
\prod_{i=1}^{M+1} (1 - \beta_i) = (1 - \beta_M) \prod_{i=1}^{M} (1 - \beta_i)
$$

$$= (1 - \beta_M) \sum_{S \subseteq [M]} \prod_{j \in S} (-\beta_j)
$$

$$= \sum_{S \subseteq [M]} \prod_{j \in S} (-\beta_j) - \beta_M \sum_{S \subseteq [M]} \prod_{j \in S} (-\beta_j)
$$

$$= \sum_{S \subseteq [M+1]} \prod_{j \in S} (-\beta_j).
$$

Setting $\beta_i = \varphi_i \left(\frac{x_i(t)}{N_i}\right)$ achieves the desired result. \qed

### 2.9 Proof of Theorem 4

**Theorem.** Given an independent multi-urn search problem, a block policy

$$u_B = (v^1, v^2, \ldots, v^{||\mathcal{V}||})$$


is optimal if and only if

\[ N_{vi} \left( \frac{2 - \varphi_{vi}}{\varphi_{vi}} \right) \leq N_{vi+1} \left( \frac{2 - \varphi_{vi+1}}{\varphi_{vi+1}} \right), \quad i = 1, 2, \ldots, |\mathcal{V}| - 1. \]

Proof. Proof of Theorem 4. First we prove that the condition in equation 2.7 implies optimality by contrapositive. Let \( u_B = (v^1, v^2, \ldots, v^{|\mathcal{V}|}) \) be a block policy that does not satisfy this condition, and let \( i \) be an index for which

\[ N_{vi} \left( \frac{2 - \varphi_{vi}}{\varphi_{vi}} \right) > N_{vi+1} \left( \frac{2 - \varphi_{vi+1}}{\varphi_{vi+1}} \right). \]

Also, we define the first stage in which urn \( v^i \) is queried in this policy as \( \tau = \sum_{j=0}^{i-1} N_j \).

We now construct an alternative block policy \( \tilde{u}_B \), so that

\[ \tilde{v}^j = \begin{cases} v^j & j \notin \{i, i+1\} \\ v^{i+1} & j = i \\ v^i & j = i + 1 \end{cases} \]

Let \( \mathbb{E}[C] = \sum_{k=0}^{N-1} \prod_{t=0}^k \mathbb{P}(w(x(t), u(t)) = 0) \) be the expected cost of policy \( u_B \) and \( \mathbb{E}[\tilde{C}] = \sum_{k=0}^{N-1} \prod_{t=0}^k \mathbb{P}(w(\tilde{x}(t), \tilde{u}(t)) = 0) \) be the expected cost of policy \( \tilde{u}_B \). For brevity, we also define

\[ \gamma = \prod_{t=0}^{T-1} P(w(x(t), u(t)) = 0) = \prod_{j=1}^{T-1} (1 - \varphi_{vi}) > 0, \] according to Corollary 4. Now consider the
difference in expected cost,

\[ \mathbb{E}[C] - \mathbb{E}[\tilde{C}] = \sum_{k=0}^{N-1} \prod_{t=0}^{\tau+N_{i}\tau + N_{i}+1-1} \mathbb{P}(w(x(t), u(t)) = 0) \]

\[ = \sum_{k=0}^{N-1} \prod_{t=0}^{\tau+\tau} \mathbb{P}(w(x(t), u(t)) = 0) \]

\[ - \sum_{k=0}^{N-1} \prod_{t=0}^{\tau+\tau} \mathbb{P}(w(\tilde{x}(t), \tilde{u}(t)) = 0) \]

\[ \gamma \left( \sum_{k=0}^{\tau+\tau} \prod_{t=0}^{\tau} \mathbb{P}(w(x(t), u(t)) = 0) \right) \]

Steps (a) and (b) follow immediately from Corollary 4. The difference in expected cost, \( \mathbb{E}[C] - \mathbb{E}[\tilde{C}] \), is strictly positive so that block policy \( u_B \) cannot be optimal. Therefore, the condition given in Theorem 4 is necessary for optimality.

Now we show that the same condition is sufficient for optimality, i.e., if a block policy satisfies equation (2.7) in Theorem 4, then it must be an optimal policy. In order to form a contradiction, suppose now that \( u_B \) is a block policy that satisfies the condition but is not optimal. Let \( u_B^* \) be the optimal block policy, which Theorem 2 guarantees to exist.

We know from the above argument that \( u_B^* \) also must satisfy the condition, which implies that policies \( u_B \) and \( u_B^* \) can only differ by permuting subsequences \( v^i, v^i+1, \ldots, v^i+\delta \) for which

\[ N_{i\delta} \left( \frac{2 - \varphi_{v^i+1}}{\varphi_{v^i}} \right) = N_{i\delta+1} \left( \frac{2 - \varphi_{v^i+\delta+1}}{\varphi_{v^i+\delta}} \right) = \cdots = N_{i\delta+\delta} \left( \frac{2 - \varphi_{v^i+\delta}}{\varphi_{v^i+\delta+1}} \right) . \]

The optimal block policy \( u_B^* \) therefore can be constructed by executing a finite number of sequential
pairwise exchanges in the urn ordering in block policy $\mathbf{u}_B$, each satisfying the condition with equality. However, it follows from the inequality in equation (2.15) that any such permutation results in the same expected policy cost. We can conclude that the expected costs of the two policies are equal, establishing the contradiction and showing $\mathbf{u}_B$ to be an optimal policy.

2.10 Proof of Theorem 6

Theorem. Given a single marble multi-urn search problem, a block policy

$$\mathbf{u}_B = (v^1, v^2, \ldots, v^{|\mathcal{V}|})$$

is an optimal policy if and only if

$$\frac{\varphi_{v^i}}{N_{v^i}} \geq \frac{\varphi_{v^{i+1}}}{N_{v^{i+1}}}, \quad i = 1, 2, \ldots, |\mathcal{V}| - 1. \tag{2.16}$$

Proof. Proof of Theorem 6. The proof is similar to that of Theorem 4. First we prove that the condition in equation (2.8) implies optimality by contrapositive. Let $\mathbf{u}_B = (v^1, v^2, \ldots, v^{|\mathcal{V}|})$ be a block policy that does not satisfy equation (2.8), and let $i$ be an index for which

$$\frac{\varphi_{v^i}}{N_{v^i}} < \frac{\varphi_{v^{i+1}}}{N_{v^{i+1}}}.$$ 

Also, we define the first stage in which urn $v^i$ is queried in this policy as $\tau = \sum_{j=1}^{i-1} N_j$.

We now construct an alternative block policy $\tilde{\mathbf{u}}_B$, so that

$$\tilde{v}^j = \begin{cases} v^j & j \notin \{i, i+1\} \\ v^{i+1} & j = i \\ v^i & j = i + 1. \end{cases}$$

Let $\mathbb{E}[C] = \sum_{k=0}^{N-1} \prod_{t=0}^{k} \mathbb{P}(w(x(t), u(t)) = 0)$ be the expected cost of policy $\mathbf{u}_B$ and $\mathbb{E}[\tilde{C}] = \sum_{k=0}^{N-1} \prod_{t=0}^{k} \mathbb{P}(w(\tilde{x}(t), \tilde{u}(t)) = 0)$ be the expected cost of policy $\tilde{\mathbf{u}}_B$. Now consider the difference in
Step (c) follows from substituting the results in Corollary 6. The difference in expected cost, 
\[ E[C] - E[\tilde{C}] = \sum_{k=0}^{N-1} \prod_{t=0}^{\tau+N_{u,i}+N_{u,i+1}-1} k \] 
\[ = \sum_{k=0}^{\tau+N_{u,i}+N_{u,i+1}-1} k \prod_{t=0}^{\tau} \mathbb{P}(w(x(t), u(t)) = 0) \] 
\[ - \sum_{k=0}^{\tau+N_{u,i}+N_{u,i+1}-1} k \prod_{t=0}^{\tau} \mathbb{P}(w(\tilde{x}(t), \tilde{u}(t)) = 0) \] 
\[ = \left( N_{u,i} - \frac{(N_{u,i} + 1)\varphi_{v,i}}{2} - N_{u,i} \sum_{j=1}^{i-1} \varphi_{v,j} + N_{u,i+1} - \frac{(N_{u,i+1} + 1)\varphi_{v,i+1}}{2} - N_{u,i+1} \sum_{j=1}^{i} \varphi_{v,j} \right) \] 
\[ - \left( N_{u,i+1} - \frac{(N_{u,i+1} + 1)\varphi_{v,i+1}}{2} - N_{u,i+1} \sum_{j=1}^{i-1} \varphi_{v,j} + N_{u,i} \varphi_{v,i+1} \right) \] 
\[ = N_{u,i} \varphi_{v,i} - N_{u,i+1} \varphi_{v,i+1} \] 
\[ = N_{u,i} N_{u,i+1} \left( \frac{\varphi_{v,i+1}}{N_{u,i+1}} - \frac{\varphi_{v,i}}{N_{u,i}} \right) > 0. \]
the two policies are equal, establishing the contradiction and showing $u_B$ to be an optimal policy. \hfill \Box

### 2.11 Proof of Theorem 7

*Theorem.* Given a multi-urn search problem and a search policy $u$, the following inequalities hold:

1. For any subset of urns $U \subseteq \mathcal{V}$ such that $u(t) \in U$,
   \[
   \mathbb{P} \left( \bigcap_{i \in U} A_i \ \bigg| \ x(t) \right) \geq \mathbb{P} \left( \bigcap_{i \in U} A_i \ \bigg| \ x(t+1) \right),
   \]
   with equality holding only in cases in which $\mathbb{P} \left( A_{u(t)} \ \big| \ x(t) \right) = 1$ or $\mathbb{P} \left( \bigcap_{i \in U} A_i \ \big| \ x(t) \right) = 0$.

2. For any stage $t$ for which urn $u(t)$ has more than one marble remaining, i.e., $N_{u(t)} - x_{u(t)}(t) > 1$,
   \[
   \mathbb{P}(w(x(t), u(t)) = 1) \leq \mathbb{P}(w(x(t+1), u(t)) = 1),
   \]
   with equality holding only when $\mathbb{P}(w(x(t), u(t)) = 1) = 0$.

*Proof.* Proof of Theorem 7. The first inequality in Theorem 7 follows from Equation 2.9 in Appendix 2.6. Given $u(t) \in U$,

\[
\mathbb{P} \left( \bigcap_{i \in U} A_i \ \bigg| \ x(t+1) \right) = \mathbb{P} \left( \bigcap_{i \in U} A_i \ \bigg| \ x(t) \right) \left( \frac{1 - \left( \frac{1}{N_{u(t)} - x_{u(t)}(t)} \right)}{1 - \left( \frac{1}{N_{u(t)} - x_{u(t)}(t)} \right) \mathbb{P}(A_{u(t)} \ \big| \ x(t))} \right) \\
\leq \mathbb{P} \left( \bigcap_{i \in U} A_i \ \bigg| \ x(t) \right).
\]

Note that if $\mathbb{P} \left( \bigcap_{i \in U} A_i \ \big| \ x(t) \right) = 0$, then $\mathbb{P} \left( \bigcap_{i \in U} A_i \ \big| \ x(t+1) \right) = 0$. Equality is likewise preserved when $\mathbb{P}(A_{u(t)} \ \big| \ x(t)) = 1$. (We intentionally omit the case when no red marble is found after drawing all of the marbles in an urn $i$ for which $\varphi_i = 1$.) Assuming $0 < \mathbb{P} \left( \bigcap_{i \in U} A_i \ \big| \ x(t) \right)$ and $\mathbb{P}(A_{u(t)} \ \big| \ x(t)) <
1, then the inequality becomes strict:

\[
0 < \left( \frac{1}{N_u(t)} - x_u(t) \right) \mathbb{P}(A_{u(t)}|x(t)) < \left( \frac{1}{N_u(t)} - x_u(t) \right) \leq 1
\]

\[
1 - \left( \frac{1}{N_u(t)} - x_u(t) \right) \mathbb{P}(A_{u(t)}|x(t)) > 1 - \left( \frac{1}{N_u(t)} - x_u(t) \right)
\]

\[
1 > \frac{1 - \left( \frac{1}{N_u(t)} - x_u(t) \right) \mathbb{P}(A_{u(t)}|x(t))}{1 - \left( \frac{1}{N_u(t)} - x_u(t) \right) \mathbb{P}(A_{u(t)}|x(t))}
\]

\[
\mathbb{P}\left( \bigcap_{i \in U} A_i | x(t) \right) > \mathbb{P}\left( \bigcap_{i \in U} A_i | x(t) \right) \left( \frac{1 - \left( \frac{1}{N_u(t)} - x_u(t) \right) \mathbb{P}(A_{u(t)}|x(t))}{1 - \left( \frac{1}{N_u(t)} - x_u(t) \right) \mathbb{P}(A_{u(t)}|x(t))} \right)
\]

\[
= \mathbb{P}\left( \bigcap_{i \in U} A_i | x(t+1) \right)
\]

To prove the second inequality in Theorem 7, first note that for any real numbers \(\alpha, \beta\), such that \(\alpha > 0\), \(\alpha + \beta > 0\), and \(|\beta| > 0\),

\[
(\alpha + \beta)^2 = \alpha^2 + 2\alpha\beta + \beta^2 > \alpha^2 + 2\alpha\beta = \alpha (\alpha + 2\beta)
\]

\[
\left( \frac{\alpha + \beta}{\alpha} \right) > \left( \frac{\alpha + 2\beta}{\alpha + \beta} \right)
\]

Now let

\[
\alpha = \sum_{S \subseteq V} (-1)^{|S|} \varphi_S \prod_{i \in S} \frac{x_i(t)}{N_i}
\]

\[
\beta = \left( \frac{1}{N_u(t)} \right) \sum_{S \subseteq V, u(t) \in S} (-1)^{|S|} \varphi_S \prod_{i \in S \setminus \{u(t)\}} \frac{x_i(t)}{N_i}.
\]

Observe that

\[
\alpha + \beta = \sum_{S \subseteq V} (-1)^{|S|} \varphi_S \prod_{i \in S} \frac{x_i(t+1)}{N_i}
\]

\[
\alpha + 2\beta = \sum_{S \subseteq V} (-1)^{|S|} \varphi_S \prod_{i \in S} \frac{x_i(t+2)}{N_i},
\]

assuming urn \(u(t)\) is queried again in stage \(t + 1\). From Lemma 1 and Corollary 2, we can conclude that \(\alpha > 0\) and \(\alpha + \beta > 0\) as long as there is a positive probability of reaching stage \(t\) without
finding a red marble. Therefore,

\[
\begin{align*}
\left(\frac{\alpha + \beta}{\alpha}\right) & \geq \left(\frac{\alpha + 2\beta}{\alpha + \beta}\right) \\
\sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{i \in S} \frac{x_i(t+1)}{N_i} & \geq \sum_{S \subseteq \mathcal{V}} (-1)^{|S|} \varphi_S \prod_{i \in S} \frac{x_i(t+2)}{N_i} \\
\mathbb{P}(w(x(t), u(t)) = 0) & \geq \mathbb{P}(w(x(t+1), u(t)) = 0) \\
\mathbb{P}(w(x(t), u(t)) = 1) & \leq \mathbb{P}(w(x(t+1), u(t)) = 1).
\end{align*}
\]

It follows from Lemma 1 that \( \beta = 0 \) only when there is no probability of drawing a red marble from urn \( u(t) \) in stage \( t \). Therefore, the inequality is strict whenever \( \mathbb{P}(w(x(t), u(t)) = 1) > 0 \). \( \square \)

### 2.12 Proof of Theorem 8

**Theorem.** Given a multi-urn search problem and a search policy \( u \), the following inequality holds:

\[
\frac{\mathbb{P}(w(x(t+1), u(t)) = 1)}{\mathbb{P}(w(x(t), u(t)) = 1)} \geq \frac{\mathbb{P}(w(x(t+1), i) = 1)}{\mathbb{P}(w(x(t), i) = 1)}.
\]

**Proof.** Proof of Theorem 8. We assume that all probabilities are positive. Note that the result in Theorem 8 can be restated

\[
\frac{\mathbb{P}(w(x(t), i) = 1)}{\mathbb{P}(w(x(t), u(t)) = 1)} \geq \frac{\mathbb{P}(w(x(t+1), i) = 1)}{\mathbb{P}(w(x(t+1), u(t)) = 1)}. \tag{2.17}
\]

Now let

\[
\begin{align*}
\gamma_i & = \sum_{S \subseteq \mathcal{V} \setminus \{u(t)\}: \ i \in S} (-1)^{|S|} \varphi_S \prod_{i \in S \setminus \{i\}} \frac{x_i(t)}{N_i} \\
\gamma_u(t) & = \sum_{S \subseteq \mathcal{V} \setminus \{i\}: \ u(t) \in S} (-1)^{|S|} \varphi_S \prod_{i \in S \setminus \{u(t)\}} \frac{x_i(t)}{N_i} \\
\gamma_{i,u(t)} & = \sum_{S \subseteq \mathcal{V} \setminus \{i,u(t)\} \subseteq S} (-1)^{|S|} \varphi_S \prod_{i \in S \setminus \{i,u(t)\}} \frac{x_i(t)}{N_i}
\end{align*}
\]
Substituting the probability distribution from Corollary 1 into equation (2.17), we have

\[
\begin{align*}
- \frac{1}{N_i} \gamma_i & - \frac{x_{u(t)}(t)}{N_i N_u(t)} \gamma_{i,u(t)} \\
- \frac{1}{N_u(t)} \gamma_u(t) & - \frac{x_i(t)}{N_i N_u(t)} \gamma_{i,u(t)} \\
- \frac{x_{u(t)}(t)}{N_i N_u(t)} \gamma_{i,u(t)} & \geq - \frac{x_{u(t)}(t) + 1}{N_i N_u(t)} \gamma_{i,u(t)} \\
0 & \leq \gamma_{i,u(t)},
\end{align*}
\]

which is true from Lemma 1. \( \square \)
Chapter 3

Finding Online Extremists in Social Networks

This chapter deals with the problem of finding extremists in online social networks. In our development that follows, we employ machine learning models to classify users and user accounts, and to model connection probabilities. Using these models and the theoretical results from Chapter 2, we develop and test an algorithm for searching for new accounts belonging to Twitter users that have been suspended.

3.1 Background and Motivation

In recent years there has been a huge increase in the number and size of online extremist groups using social networks to harass users, recruit new members, and incite violence. These groups include terrorist organizations such as the Islamic State of Iraq and Syria (ISIS) [12], white nationalists and Nazi sympathizers [136], and “cyberbullies” who target individuals with offensive and harassing messages [26]. Of particular concern is the danger posed to public safety by terrorist groups. The threat from online terrorist groups such as ISIS has become so severe that U.S. president Barack Obama recently said “The United States will continue to do our part, by working with partners to counter ISIL’s\(^1\) hateful propaganda, especially online” [69]. It is suspected that the online presence of ISIS may have been responsible for radicalizing individuals and motivating them to commit acts of terror [23].

Social networks have recently begun taking actions to actively combat online extremists. For

\(^1\)ISIL is another name for ISIS and stands for is the Islamic State of Iraq and the Levant.
instance, Twitter, which has become the main venue for ISIS users to spread their propaganda [69], has been very aggressive in its response to ISIS. In August 2016, Twitter reported that it had shut down over 360,000 ISIS accounts and its daily suspensions of terrorism-linked accounts have jumped 80 percent since 2015 [1]. Twitter identifies extremist accounts primarily based on reports from its users, but it has begun using proprietary spam-fighting tools to supplement these reports. These tools have helped to automatically identify more than one third of the accounts that were ultimately suspended for promoting terrorism on Twitter [137].

The efforts of social networks such as Twitter have been effective at limiting the reach of online extremist groups such as ISIS. However, not all extremist users are shut down and they are constantly returning to the social network after being suspended. In addition, much of the success in mitigating the threats of extremist groups has relied upon the cooperation of the social networks themselves. For instance, Twitter has dedicated teams to review user reports of potential extremist accounts [1]. However, if extremist users migrate to other social networks, there is no guarantee that the companies which operate these networks will be as cooperative or dedicate as many resources to dealing with online extremists. Therefore, what is needed is a set of capabilities that can be used by authorities to combat online extremists which do not rely upon the cooperation of social network operators and can be applied to any social network.

### 3.1.1 Overview of Contributions

The case of ISIS in Twitter is useful to understand general behavioral patterns of online extremist users in social networks. We use these behaviors to guide the development of capabilities for combating online extremists in general social networks. These capabilities include identifying new extremist users and finding these users as they continue to return to the social network after having their accounts suspended. We provide a detailed analysis of these behaviors and develop the corresponding capabilities in Sections 3.3, 3.4, 3.5, and 3.6. Here we will provide a concise overview of our major contributions, in particular the different behavioral patterns of online extremists and the corresponding capabilities we develop.

**Suspensions.** Online extremist users post content which violate the Terms of Use of social networks, leading to the suspension of their accounts. These suspensions occur in response to user reports, but many social networks are beginning to use algorithms to automatically detect any violative content [137]. Going one step further, it would be useful to have a capability to flag users as potential extremists before they post any content at all. There are potential features of
an account that may predict if it belongs to an extremist user. For instance, the account may not publicly declare its geographical location. Also, the users to which the account connects may indicate whether or not the account belongs to an extremist user. In Section 3.3 we use these intuitions to develop a method to automatically predict if an account will be suspended without requiring it to post any content.

**Creating Multiple Accounts.** After being suspended, online extremist users will quickly create new accounts and continue their activities on the social network. This makes it difficult to keep an extremist user off the social network. Typically the new account resembles the suspended account in several aspects. For instance, the names and profile pictures may be very similar. A useful capability would be the ability to identify if multiple accounts as belong to the same user. This would allow for more accurate monitoring and tracking of extremist users. We develop such a capability in Section 3.4.

**Re-following Previous Friends.** A user in a social network generally follows a set of users. In Twitter these followed users are referred to as the **friends** of the user and the user is referred to as their **follower**. Upon returning to the social network after being suspended, an extremist user will generally re-follow some of his previous friends. If we knew which previous friends a suspended user re-follows, this information could be used to find the user’s new account in the social network. There may be features of the friends which make it more likely the suspended user will re-follow them. In Section 3.5 we use these features to develop a method to predict who suspended users re-follow.

**Suspended User Search.** Authorities may wish to find suspended users when they return to a social network. The operator of the social network is notified every time a new user enters the network and can use our account matching capability to see if the new user matches a previously suspended user. However, if one is not the operator of the social network, then one must search the network to see if the suspended user has returned. Because of the size of the social network, this search could require a large amount of time and resources. To overcome this challenge, we develop an efficient network search policy in Section 3.6 based on a variant of a Pólya’s urn model which utilizes our re-following prediction capability from Section 3.5. We show on real data from ISIS users in Twitter that our search policy can substantially reduce the time needed to find returning extremist users in social networks.

The remainder of this chapter is organized as follows. We review the extant literature relevant to our work in Section 3.1.2. We provide a detailed overview of the data used for our analysis in
Section 3.2. Section 3.3 presents our predicting suspensions capability. We present our account matching capability in Section 3.4. Our method for predicting re-following is presented in Section 3.5. Section 3.6 details our model for network search and an optimal search policy. We conclude in Section 3.7.

3.1.2 Previous Work

**Analysis of Online Extremist Networks.** There are several studies focused on ISIS users in social networks. One of the earlier studies, Berger and Morgan [12], characterizes the number, behavioral traits, and organization of Twitter ISIS users. A subsequent study by the same authors found that the reach of ISIS had been limited by the beginning of 2016 due to the efforts of Twitter to suspend ISIS accounts [14]. Johnson et al. [75] study the dynamics of ISIS users in the Russian social network VKontakte and suggest that shutting down smaller pro-ISIS groups can prevent the emergence of larger, more influential groups. Ferrara et al. [52] develop models to predict which users will be suspended for being in ISIS, who will retweet ISIS content, and who will interact with ISIS users. This work is similar to our work, but the authors do not study many of the capabilities we develop such as identifying multiple accounts from a single user, re-following old friends, or searching for suspended users.

There have also been several works looking at identifying extremist content in groups beyond ISIS. Scanlon and Gerber [125] develop methods to automatically classify content that is used for recruiting members to extremist groups. Similar work by Sureka and Agarwal [134] uses machine learning to detect content that promotes hate and extremism. Machine learning methods have also been used to detect cyberbullies based on the content they post [119, 44]. Dinakar et al. [45] build upon this work to develop an approach for mitigating the threat of cyberbullying.

**Spam/Bot Detection.** Closely related to our capabilities on predicting suspensions is the work done on detecting online bots (non-human users) or malicious users. Several approaches have been developed which use different types of behavioral features. The type of content (URL’s, user mentions) was found to be predictive of Twitter bots by Lee et al. [89]. Temporal behavior and aggregate network properties (in-degree, out-degree) were used to identify Twitter bots by Chu et al. [33]. Dickerson et al. [43] demonstrate that the sentiment of the posted content can be used to identify bots. All of these approaches are designed to detect automated behavior. However, they may not be as effective for human users who engage in extremist behavior. Also, many of these approaches require the user to post some content in order to detect whether or not they are bots. An
approach related to ours was used by Kumar et al. [87], which relied purely on network structure to identify malicious users in social networks where edges have a polarity (friend/enemy). In contrast to this extant work, our approach combines both behavioral features with refined network features to detect extremist users.

**Network Search.** Our network search problem is similar to those presented by Alpern and Lidbetter [5] and Dagan and Gal [38], who have done much work in this area. Unlike their work, in which the searcher and the target are assumed to be operating in a physical network, our problem of searching a social network admits a different set of search constraints. In our network search problem, the searcher is not constrained to move along edges. Instead, the searcher can examine the neighbors of any of a set of nodes that are known to him, but each of these queries comes at a cost. This alternative representation of network search follows from one of the original search problems posed by Black [19], in which a searcher looks for the search target among a set of possible locations. Each location has a known probability of containing the target and a known probability of finding the target, if it is there. Our network search application adapts this simple search model to a network setting. Instead of limiting the search target to be at at most one of a set of possible locations, in a network search the target could be connected to more than one of the nodes known to the searcher. Also, the method of querying the neighbors of a node causes the probability of finding the target to change with each observation.

Our network search model builds directly on the multi-urn search model developed in Chapter 2. However, in this work the major difference is that we allow for more than one query to be done in each step, which results in slight differences in the optimal policies.

### 3.2 Data

The data we study in this work comes form the micro-blogging site Twitter [141]. Twitter serves as a front line public platform used by ISIS for outreach and recruitment. ISIS’s presence on Twitter, and its consistent success at gaining support and recruits through the social media site has been deeply analyzed and well-documented [12].

The nature of Twitter as a social media platform was discussed in Chapter 1. Twitter users form a social network by connecting to each other. This network is directed and this directionality dictates the flow of information. A user forms a connection with someone on Twitter by *following* him or her. Each account a user is following is known as this user’s *friend* and the user is known
as the friend’s follower.

For this research, we collected Twitter data from approximately 5,000 “seed” users, who were either known ISIS members or who were connected to many known ISIS members as friends or followers. The names of these seed users were obtained through news stories, blogs, and reports released by law enforcement agencies and think tanks [82]. The data was collected at various times throughout the calendar year 2015, using Twitter’s REST API (see [138]).

For each seed user we collected the user account profile information, including the screen name, name, description, location, profile picture, and profile banner at the time of the collection. We also obtained the user account ID number, which is the only unchangeable unique account identifier. In addition to obtaining seed users’ profile information, we collected the same set of profile information for each seed user’s friends and followers. As a result the number of user profiles contained in the data set grew to over 1.3 million.

We downloaded all publicly available tweets from each seed user’s timeline at the time of collection. For each tweet we obtained the unique tweet ID assigned by Twitter, the tweet text, the time of the post, all hashtags, user mentions, URLs, and images contained in the tweet, and whether the tweet was a retweet of or reply to another tweet. The total number of tweets in our data is approximately 4.8 million.

Finally, we tracked many of the accounts for several months in 2015 in order to see if they were ever suspended. We do not know the reason for suspension, but given that these accounts were associated with known ISIS users, we assume the suspension was related to some form of extremist propaganda that violated Twitter’s user agreement. We tracked all of the user accounts collected in June, 2015, including the seed accounts and their friends’ and followers’ accounts. This data set includes 646,961 accounts in total, of which 35,080 (or 5.4%) had been suspended as of September 23, 2015.

### 3.3 Predicting Account Suspensions

The first capability we develop to combat online extremism is to predict which accounts belong to new extremist users. In this section we develop an approach to this using logistic regression. We label any account in our data set as extremist if it was suspended by Twitter. Therefore, to detect extremists we predict which accounts are suspended by Twitter. We accomplish this using a logistic regression model based upon features of the user accounts. We provide out-of-sample performance
evaluation of the model and provide insights on what factors might be useful in predicting whether a Twitter user is going to be suspended for violative behavior.

To train, validate, and test this prediction model we use a subset of the accounts whose suspension status we tracked. We randomly selected two non-overlapping samples of this data set, each consisting of 5,000 accounts and maintaining the 5.4% suspension rate, which is the overall suspension rate of these accounts. These data sets were used for training and validation.

For our logistic regression model, the response variable is whether the account was still active as of September 23, 2015. The predictors are obtained from the wide array of information associated with the user accounts. Some of these relate to the account itself, while others have to do with the network connections of the account. The variables used as predictors for our model are listed in Table 3.1. While we have observed that the number of screen name changes associated with a user account might serve as a good predictor of future suspension, we assume that this information is not necessarily known for an arbitrary account we wish to classify. Similarly, we assume we do not know if the account was following accounts that were suspended in the past. All features we use are what could be measured for a new account that has not been seen before.

Table 3.1: Features for predicting Twitter suspensions.

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Following each of 2,376 active ISIS seed accounts in our data (2,376 binary variables).</td>
</tr>
<tr>
<td>Account</td>
<td>Date and time the account was created (numeric).</td>
</tr>
<tr>
<td>Account</td>
<td>Number of “friends” and “followers” connected to the account (2 numeric variables).</td>
</tr>
<tr>
<td>Account</td>
<td>Number of tweets from the account (numeric).</td>
</tr>
<tr>
<td>Account</td>
<td>Geo-location enabled (binary).</td>
</tr>
<tr>
<td>Account</td>
<td>“Protected” account (posts are not visible to the public) (binary).</td>
</tr>
<tr>
<td>Account</td>
<td>Verified account (identity confirmed by Twitter) (binary).</td>
</tr>
</tbody>
</table>

3.3.1 Results

We fit a logistic regression model with $L_1$-norm regularization to the training data. From validation, we find that setting the regularization constant to 10 consistently provided near-optimal performance. The resulting coefficient estimates were nonzero for 89 of the predictor variables, of which 81 corresponded to following certain accounts. The signs and magnitudes of the coefficients give us some idea of the effects of some of the predictor variables. The coefficient estimates indicate that accounts that had enabled geotagging and accounts that had Twitter-verified owners were
Figure 3-1: ROC curve for the regularized logistic regression classifier for Twitter suspensions ($P_F$ is the false detection rate and $P_D$ is the true detection rate).

much less likely to be suspended. This is not surprising given that we expect online extremists to want to mask their identity and location. The effects of friendships were less intuitive and difficult to interpret. In total we found that 38 accounts had a positive sign and 43 had a negative sign. However, there was no clear pattern that we could find among the positive sign accounts or negative sign accounts. More detailed analysis may reveal what made following these accounts increase or decrease the likelihood of suspension. Nonetheless, just knowing the value of the regression coefficient was sufficient to predict suspensions.

Figure 3-1 shows the receiver-operator characteristic (ROC) curve on the validation set and on the test data, which was comprised of the 636,961 accounts not used for training and validation. The area under the curve (AUC) on the test data is approximately 0.83. We can see from the curve that we can detect about 60% of suspended users in the test set with only a 10% “false positive” rate. This efficiency occurs by setting a classification probability of 0.1, i.e., classifying an account as an ISIS account if the regression function assigns a probability of suspension higher than this threshold.
It is important to note that because 94.6% of the accounts in our data were not suspended, a 10% false positive rate represents a greater number of false positive classifications than true detections using this logistic regression model. However, it is also important to consider that accounts that have not been suspended could still be suspended in the future, or that some accounts in our data should be suspended but have succeeded in avoiding detection.

Table 3.2: Summary of sampled accounts from those incorrectly classified as suspensions using the regularized logistic regression model.

<table>
<thead>
<tr>
<th>Screen Name</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@abdulnagi313</td>
<td>Few tweets, difficult to discern nature of account.</td>
</tr>
<tr>
<td>@445468a7e3fc45c</td>
<td>Very few tweets, user apparently follows ISIS activity and members on Twitter; possibly conducting research or surveillance.</td>
</tr>
<tr>
<td>@613780</td>
<td>Tweets Quranic verses in Arabic every few hours in consistent format; likely a Twitter bot.</td>
</tr>
<tr>
<td>@aarishmajeed</td>
<td>Account with no tweets following three ISIS-related media accounts.</td>
</tr>
<tr>
<td>@men9174</td>
<td>Arabic-language pornography account followed by one of our seed accounts; following many other pornographic accounts.</td>
</tr>
</tbody>
</table>

Sampling from the false positives resulting from this classification returned some accounts that were clearly ISIS supporters, supporting this notion that many accounts should be or soon would be suspended. Many of these “false positives,” however, were ISIS researchers, media, or otherwise difficult to discern. Table 3.2 provides a summary of five randomly selected false positives found in our test data, when applying the classification probability threshold of 0.1. The inclusion of the pornographic account @men9174 as a false positive is interesting and concerning. Investigation reveals that this account is not following any of our ISIS seed accounts. Our model classified this account with a probability of 0.101, very near our threshold, based primarily on its profile features.

3.4 Detecting Multiple Accounts

Now that we have a model for predicting suspensions, the next question we address is whether we can automatically determine whether two accounts belong to the same user. This question is relevant because we have observed many cases in which a user simply creates a new account after being suspended. We have even found ISIS accounts dedicated to the purpose of broadcasting suspended
users’ new accounts to ISIS members and supporters. By detecting multiple accounts belonging to the same user, one can prevent extremist users from restarting their violative behaviors by creating new accounts and effectively keep them suspended from the social network.

Twitter profiles essentially serve as avatars; the syntax and pictures provide cues about the identity of the account holder. This is true for ISIS users as well and is intrinsic to the tactic behind the ISIS-based networks directed at recruitment. As a result, when a suspended user opens a new account in Twitter, we have been able to identify it by comparing the names, images, screen names, and descriptions associated with each account. This behavior is intuitive: these newly created accounts include cues to permit the suspended user’s followers to identify and re-follow the recreated accounts. We have found many examples of this predictable reiteration of account profile features in our data.

3.4.1 Suspended User Behavior

In addition to regular account suspensions, we also observed that known ISIS users in our data set changed their screen names regularly. We hypothesize that frequent screen name changes provide a means of avoiding tracking and detection, while retaining account information, friends and follower connections, and Twitter posts. We also note that accounts that exhibit multiple screen name changes had higher suspension rates, which could mean that users are changing their screen names to avoid suspension.

Table 3.3 provides a timeline of screen name and name changes for two such accounts, purportedly belonging to British citizen Sally Jones who had adopted the online alias “Umm Hussain al-Britani,” [129]. Sally Jones and her husband, Junaid Hussain achieved celebrity status in ISIS, primarily due to Junaid Hussain’s role in creating and leading the “CyberCaliphate,” as well as his previous involvement in the “Team Poison” hacking group. Junaid Hussain was killed in a US air strike in August 2015 [2]. The timeline in Table 3.3 was reconstructed from observed tweets, but the tweets from both of these accounts are no longer available due to account suspensions. We note that the screen name changes became much more frequent when the user believed her behavior might result in suspension. We also observe that in almost all cases, the user chooses some variant of the same online handle, e.g., “OumHu55inBrit,” which helps her retain her online identity and signals her status by announcing her attachment to Junaid Hussain, who always used the online alias “Abu Hussain al-Britani” (see follow-on discussion and Table 3.7).

While there might have been additional screen names and tweets associated with these accounts
that we did not capture, we found the type of online behavior exhibited in Table 3.3 is indicative of many of the ISIS-supporting accounts in our data set. Following suspension, the user apparently opens a new account and continues the same tactic, all the while adopting very similar account screen names and names. Prominent ISIS members Sally Jones and Junaid Hussain provide examples of this behavior; accounts associated with them appear frequently in our ISIS data. Querying our data for user accounts with a name similar to “Umm Hussain Al-Britani” returns 23 distinct entries, all of which have been suspended.

Empirically, we observed screen name changes in approximately 10% of the accounts in our data that were eventually suspended, while in the accounts that remained active the number was close to 1%. Furthermore, anecdotal investigation of active accounts with multiple screen name changes suggests that many of these accounts are also ISIS-related. Figure 3-2 provides a histogram comparison of the number of screen names associated with active and suspended accounts in our data set. It is clear from the figure that the suspended accounts are much more likely to have more screen names. For example, even though active accounts make up over 94% of our data, only 18% of the accounts with over 20 unique screen names are still active.

These observations motivated our development of a method for locating new accounts belonging

Table 3.3: Partial screen name—tweet timelines for two Twitter user accounts purportedly belonging to Sally Jones. These accounts have been suspended by Twitter and are no longer available.

<table>
<thead>
<tr>
<th>Tweet Time</th>
<th>Screen Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-09-30 11:45:37</td>
<td>OumHu554inBrit</td>
</tr>
<tr>
<td>2015-09-30 19:58:15</td>
<td>OumHu554inBrit</td>
</tr>
<tr>
<td>2015-10-02 13:43:59</td>
<td>Mrsl337</td>
</tr>
<tr>
<td>2015-10-02 21:28:54</td>
<td>OumHu554inBrit</td>
</tr>
<tr>
<td>2015-10-03 00:48:01</td>
<td>UmmHu55ain2</td>
</tr>
<tr>
<td>2015-10-03 15:30:08</td>
<td>Oum1337</td>
</tr>
<tr>
<td>2015-10-03 16:52:39</td>
<td>OumHu554inBrit</td>
</tr>
<tr>
<td>2015-10-03 16:55:45</td>
<td>OumHu554inBrit</td>
</tr>
<tr>
<td>2015-10-03 17:24:06</td>
<td>OumHu554inBrit</td>
</tr>
<tr>
<td>2015-10-03 23:31:29</td>
<td>UmmHussain9ll</td>
</tr>
<tr>
<td>2015-10-04 13:20:47</td>
<td>OumHu554inBrit</td>
</tr>
</tbody>
</table>

†In this tweet the user warns she is about to release information that could get her suspended, and encourages her followers to be ready to retweet her.
‡This is the first tweet in a new user account, as the previous one was suspended.
to previously suspended users. The first step in this process was to develop an automated method of identifying whether a pair of accounts belong to the same user. To achieve this pairwise classification, we employ a supervised machine learning approach, which is described next.

3.4.2 Profile Comparison Metrics

We define a Twitter user profile as a vector of profile features $\mathbf{x}$ associated with a Twitter account. A Twitter account can only have a single user profile at any point in time. The features of the profile are not fixed, however. As we have noted, cases exist in our data in which users changed their screen name or other profile features, resulting in our obtaining multiple user profile feature vectors belonging to the same account.

While it is possible for a single Twitter account to belong to different users at different times (e.g., an account gets hacked or one user simply provides the account login information to another), we assume that all of the profiles associated with the same Twitter account belong to a single user. Our classification goal is therefore to compare two user profiles $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ from different Twitter accounts and identify whether or not they belong to the same user.

In order to train a model to perform this classification, we must construct profile comparison features from profile pairs $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ that are useful in establishing whether they belong to the same
user. Building on our qualitative observations of individual ISIS Twitter users retaining identifying similarities between their multiple user profiles, we propose a set of similarity metrics based on comparisons of the following four profile features: screen name, user name, profile picture, and profile banner image. These similarity metrics are based on user profile characteristics that are publicly available on all accounts, even if the user has “protected” the account using Twitter’s privacy settings.

**Screen Name and User Name Similarity Metrics**

In comparing two screen names or two user names, we use the well-known Levenshtein ratio (see [130]) to provide a measure of distance between two strings. This ratio involves counting the number of character additions, deletions, or place exchanges required to transform one string into the other. This number is normalized by the length of the longer string and then subtracted from one. If we let $S$ be a set of strings of various lengths, the Levenshtein ratio can be thought of as a function $L : S^2 \rightarrow [0, 1]$ where $L(s_1, s_1) = 1$ and $L(s_1, s_2) = L(s_2, s_1)$ for any $s \in S$. $L(s_1, s_2) = 0$ implies that strings $s_1$ and $s_2$ are not at all similar.

Our first two comparison features, $\phi_1$ and $\phi_2$, are simply the screen name and user name Levenshtein ratios:

$$\phi_1(x^{(i)}, x^{(j)}) = L(x^{(i)}_{SN}, x^{(j)}_{SN}), \phi_2(x^{(i)}, x^{(j)}) = L(x^{(i)}_N, x^{(j)}_N),$$

where $x^{(i)}_{SN}$ and $x^{(i)}_N$ denote the respective screen name and user name of account profile $x^{(i)}$. Figure 3-3 provides an illustration of five screen name pairs and their corresponding Levenshtein ratios.

![Example Screen Name Levenshtein Ratios](image)

**Profile Picture and Profile Banner Similarity Metrics**

We employ a simple image average hash algorithm (e.g., [115]) to compare two pictures. Essentially, the algorithm partitions the image into $8 \times 8$ equal-sized rectangular sub-images and then identifies
whether the average shade of each sub-image is brighter or darker than the overall image average. The algorithm runs efficiently and returns an $8 \times 8$ binary matrix, which can easily be represented as a non-negative integer.

We denote the hash algorithm as a function $H : \Psi \rightarrow \mathbb{Z}_+$, where for any $\psi_1, \psi_2 \in \Psi$,

$$
\psi_1 = \psi_2 \Rightarrow H(\psi_1) = H(\psi_2)
$$

$$
H(\psi_1) \neq H(\psi_2) \Rightarrow \psi_1 \neq \psi_2
$$

$$
H(\psi_1) = H(\psi_2) \Rightarrow \psi_1 \approx \psi_2.
$$

Two images with the same hash value contain very similar patterns of shade. Therefore, we assume that images with the same value are the same image. Our image similarity metric $h$ is a simple step function that follows from this assumption:

$$
h : \Psi^2 \rightarrow \{0, 1\}, \quad h(\psi_1, \psi_2) = \begin{cases} 
1 & H(\psi_1) = H(\psi_2) \\
0 & H(\psi_1) \neq H(\psi_2).
\end{cases}
$$

We use this image similarity metric to construct our third and fourth features:

$$
\phi_3(x^{(i)}, x^{(j)}) = h(x^{(i)}_{PP}, x^{(j)}_{PP})
$$

$$
\phi_4(x^{(i)}, x^{(j)}) = h(x^{(i)}_{BP}, x^{(j)}_{BP}),
$$

where $x^{(i)}_{PP}$ and $x^{(i)}_{BP}$ are the respective profile and banner pictures for profile $x^{(i)}$. These features are simply binary indicators for whether or not the images being compared have the same average hash matrix.

### 3.4.3 Data Set Construction

Having defined pairwise account profile similarity features, our next step was to clean the data and extract the features for use in a classification model. Initially we examined 4,339 seed user accounts collected before June 4, 2015. However, in order to keep the string similarity metrics consistent, we removed 395 accounts with user name strings that did not use the Latin alphabet. This left us with 3,944 user profiles. Within this set, we knew some user profiles we collected belonged to the same Twitter account and therefore the same user. These accounts were identifiable by the
Twitter user ID, which does not change even if a user changes his or her screen name or other profile features. Our set of 3,944 profiles contained 3,855 unique Twitter accounts (i.e., unique user IDs), corresponding to 3,855 seed users. For each pair of user profiles \((i, j)\), we computed a feature vector \(\phi^{(i,j)}\) of the four similarity metrics, resulting in \(\binom{3,944}{2} = 7,775,596\) pairs.

3.4.4 Data Labeling

We assume that each pair of user profiles either belong to the same user or belong to different users. We denote this classification with binary class variable \(y^{(i,j)}\), where

\[
y^{(i,j)} = \begin{cases} 
1 & \text{Profiles } i \text{ and } j \text{ belong to the same user} \\
0 & \text{Profiles } i \text{ and } j \text{ belong to different users.}
\end{cases}
\]

Of the 7,775,596 pairwise profile comparisons in our data, 95 could be traced to the same user because they actually belonged to the same account, identifiable by the Twitter user ID. Although we do not seek to classify profiles belonging to the same account because we can already assume they belong to the same user, we left these comparison points in the data set as labeled data in order to train the classification model. Updating profile features for an existing account is a different action than creating a new Twitter account, however, causing this labeled set to be biased toward profiles that are very similar. On the other hand, when a user creates a new Twitter account, he or she must deliberately set or leave blank each of the profile settings. As a result, we do not expect the same level of similarity between two user profiles associated with separate accounts but belonging to the same user, when compared to the similarity between two profiles belonging to the same Twitter account. For this reason, using these 95 pre-labeled data points for training might not be very useful for our purpose.

We also do not have any points classified as accounts belonging to different users. To solve this problem, we labeled a subset of comparisons in our data set using the following method.

1. If profile \(x^{(i)}\) and profile \(x^{(j)}\) share the same user ID, set label \(y^{(i,j)} = 1\). These are the 95 profile comparisons that are known to belong to the same user.

2. If profile \(x^{(i)}\) and profile \(x^{(j)}\) do not share the same user ID, and

\[
\langle x^{(i)}, x^{(j)} \rangle : \| \phi^{(i,j)} \|_2 < 0.1, \quad (3.1)
\]
we set label $y^{(i,j)} = 0$. These conditions establish that the profiles have very little in common, so we assume they belong to different users. Table 3.4 provides an example of the features associated with a pair of accounts meeting this criterion.

3. We manually label a randomly selected subset of unlabeled pairs that exhibit relatively high similarity metrics. We chose 168 pairs where

$$\| \phi^{(i,j)} \|_2 > 0.85,$$

for manual labeling. In assigning a label to these pairs, we considered all available data in comparing the two profiles, including Twitter posting habits and account profile features, such as location and description, that are not considered in the model. We found that 82 of these pairs were accounts belonging to the same user, while the remaining pairs were from different users. Table 3.5 provides an example comparison of the features of a pair of accounts meeting this criterion.

Table 3.4: Accounts exhibiting very low similarity, according to the selection criterion given in equation (3.1).

<table>
<thead>
<tr>
<th>Feature (k)</th>
<th>User $i$</th>
<th>User $j$</th>
<th>$\phi^{(i,j)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>2683126250</td>
<td>3108319204</td>
<td>[NA]</td>
</tr>
<tr>
<td>Screen Name</td>
<td>khalibinalwale</td>
<td>profomar0</td>
<td>0.08</td>
</tr>
<tr>
<td>Name</td>
<td>Abu Muslim</td>
<td>prof</td>
<td>0.00</td>
</tr>
<tr>
<td>Profile Picture</td>
<td>00...c3</td>
<td>09...cc</td>
<td>0.00</td>
</tr>
<tr>
<td>Profile Banner</td>
<td>00...00</td>
<td>[None]</td>
<td>0.00</td>
</tr>
</tbody>
</table>

$\| \phi^{(i,j)} \|_2 = 0.08$

Table 3.5: Accounts exhibiting very high similarity, according to selection criterion given in equation (3.2). These accounts were manually labeled as belonging to the same user, i.e., $y^{(i,j)} = 1$.

<table>
<thead>
<tr>
<th>Feature (k)</th>
<th>User $i$</th>
<th>User $j$</th>
<th>$\phi^{(i,j)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>3307258107</td>
<td>3297609231</td>
<td>[NA]</td>
</tr>
<tr>
<td>Screen Name</td>
<td>Ahmes_Zirve_</td>
<td>Ahmes_Zirve</td>
<td>0.88</td>
</tr>
<tr>
<td>Name</td>
<td>Ahmes Zirve</td>
<td>Ahmes Zirve</td>
<td>1.00</td>
</tr>
<tr>
<td>Profile Picture</td>
<td>ff...ff</td>
<td>ff...ff</td>
<td>1.00</td>
</tr>
<tr>
<td>Profile Banner</td>
<td>[None]</td>
<td>[None]</td>
<td>1.00</td>
</tr>
</tbody>
</table>

$\| \phi^{(i,j)} \|_2 = 1.94$
### 3.4.5 Classification Model

From our set of labeled data, we set aside 10% for out of sample evaluation of model performance. This percentage was enforced for each of the three labeling methods, so that the test set included 10% of the hand labeled data points, for example. We then fit an $L_1$-regularized logistic regression model on the training data. In other words, we assume

$$\mathbb{P}(y^{(i,j)} = 1) = \left(1 + e^{\beta^T \phi^{(i,j)} + \beta_0}\right)^{-1}$$

We identified $\lambda = 10$ as the regularization parameter that provided the best performance in cross validation. The intercept and coefficients for the logistic regression model fit on the training data are shown in Table 3.6. Interestingly, profile banner similarity is not useful in this model in determining the probability of two profiles belonging to the same user.

#### Table 3.6: Regression coefficients for matching accounts.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Regression coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-8.05</td>
</tr>
<tr>
<td>Screen name Levenshtein ratio ($\phi_1$)</td>
<td>2.94</td>
</tr>
<tr>
<td>User name Levenshtein ratio ($\phi_2$)</td>
<td>7.05</td>
</tr>
<tr>
<td>Profile picture hash matrix ($\phi_3$)</td>
<td>1.88</td>
</tr>
<tr>
<td>Banner picture hash matrix ($\phi_4$)</td>
<td>0</td>
</tr>
</tbody>
</table>

The receiver-operator characteristic (ROC) curve plotted for the manually labeled training and test data combined is given in Figure 3-4. ROC curves plotted separately for the training and test data were very similar, and classification on the test data points that were not manually labeled (i.e., they were labeled using steps (1) or (2) of the labeling method given in section 3.4.4) was nearly perfect. The AUC in Figure 3-4 is approximately 0.92.

We view the ROC curve on the manually labeled data in Figure 3-4 as an approximation for the “worst case” performance of the classifier. We selected these pairs for manual labeling because they exhibited some degree of similarity, based on the $L_2$ norm of the comparison feature vector, anticipating that they would be among the most difficult points to classify. As noted previously, plotting the ROC curve on all of the labeled data, or on the entire test set, shows near perfect classification.

Because we anticipate that most account pairs belong to different users, maintaining a low false positive misclassification rate is important. A small false positive rate could equate to a large number of misclassified points. For this reason, we select a false positive threshold of 2% on the
Figure 3-4: Logistic regression ROC curve on hand labeled data ($P_F$ is the false detection rate and $P_D$ is the true detection rate).

hand-labeled ROC curve. This threshold leads us to a classification probability threshold of 0.782, as indicated in Figure 3-4. In other words, we assign a classification $\hat{y}(i,j)$ to a profile pair $(x^{(i)}, x^{(j)})$ according to the function

$$\hat{y}(i,j) = \begin{cases} 
1 & \left(1 + e^{\beta T \phi(i,j) + \beta_0}\right)^{-1} \geq 0.782 \\
0 & \left(1 + e^{\beta T \phi(i,j) + \beta_0}\right)^{-1} < 0.782.
\end{cases} \tag{3.3}$$

Based only on the hand-labeled data ROC, we expect this classifier to correctly identify over 70% of account pairs belonging to the same user while misclassifying less than 2% of the account pairs belonging to different users. Because the manually labeled data consists of account pairs that exhibit some substantial measure of similarity, we expect performance on the entire data set to be much better, similar to the near-perfect classification on the test data.

When we apply the classifier in equation (3.3) to the entire data set, we obtain 318 account pairs classified as belonging to the same user. Sixty-two of these pairs have the same account ID and are therefore known to be from the same account, while the remaining 256 pairs come from different Twitter accounts. Figure 3-5 provides a network representation of these account connections. Each node in the plot represents a unique Twitter account. An edge drawn between two accounts indicates
Figure 3-5: Graph representation of accounts belonging to the same user using our regression model and equation (3.3) with a threshold of 0.782.

our classification equation labels the pair of accounts as belonging to the same user. Only accounts with at least one edge are depicted in Figure 3-5.

Most of the components in the graph depicted in Figure 3-5 are fully connected, which is as we would expect. Component A is an example of a fully connected component, consisting of five Twitter accounts. These account profile features are listed in Table 3.7. They are all very similar and indeed appear to belong to the same user.

Component B, on the other hand, consists of three accounts but is not fully connected. Table 3.8 provides a list of the profile features associated with these three accounts. While they all appear to belong to the same user, comparison of the first and third profiles given in the table resulted in probability $P(y^{(i,j)} = 1) = 0.774$, which falls below our classification threshold. While in this case these two accounts are connected by way of a third account that meets the classification threshold with both of them, it is clear that setting threshold this high does indeed miss some pairs of accounts that probably do belong to the same user. We discuss the sensitivity of the results as a function of the classification threshold in Section 3.4.6.
Table 3.7: Accounts comprising component A. While average hash values for profile pictures are abbreviated, they are the same for all profiles.

<table>
<thead>
<tr>
<th>Screen Name</th>
<th>Name</th>
<th>Profile Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlJabarti28</td>
<td>Abu Yusuf Al-Jabarti</td>
<td>20...00</td>
</tr>
<tr>
<td>BanuKombe</td>
<td>Abu Yusuf Al-Jabarti</td>
<td>20...00</td>
</tr>
<tr>
<td>enkorela</td>
<td>Abu Yusuf Al-Jabarti</td>
<td>20...00</td>
</tr>
<tr>
<td>ouaicheu</td>
<td>Abu Yusuf Al-Jabarti</td>
<td>20...00</td>
</tr>
<tr>
<td>ouaisheu</td>
<td>Abu Yusuf Al-Jabarti</td>
<td>20...00</td>
</tr>
</tbody>
</table>

Table 3.8: Accounts comprising component B.

<table>
<thead>
<tr>
<th>Screen Name</th>
<th>Name</th>
<th>Profile Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aqidahhaqq</td>
<td>Colonel Shaami</td>
<td>[None]</td>
</tr>
<tr>
<td>AnsarAlUmmah49</td>
<td>Colonel Shaami</td>
<td>[None]</td>
</tr>
<tr>
<td>buruan8</td>
<td>Colonel Shaami</td>
<td>[None]</td>
</tr>
</tbody>
</table>

3.4.6 Classification Threshold Sensitivity

We provide a brief discussion of the sensitivity of the results to changes in the classification threshold. In the previous section, we selected threshold $P = 0.782$ based on the shape of the ROC curve and our desire to keep the number of false positive classifications low. We now consider how different values of threshold $P$ affect the “paired accounts” graph depicted in Figure 3-5.

Figure 3-6 gives several properties of the “paired account” graph as a function of $P$. As we would expect, when our classification threshold $P = 0$ the graph is fully connected, which indicates that all accounts are classified as belonging to the same user. As $P$ increases, the number of connected accounts and the size of the giant component decrease rapidly. Of interest is the transitivity, measured on the right-hand scale in Figure 3-6. If we had access to the true classifications so that we could produce a graph of connected accounts that belonged to the same users, each component would be fully connected. Network transitivity provides measure of how much a graph exhibits this property by computing the fraction of connected triads that are fully connected.

We see from Figure 3-6 that transitivity steadily increases as we increase the probability threshold, eventually peaking near 1 as $P$ increases beyond approximately 0.8. At this point, the components in the resulting paired account graph are all completely connected or nearly completely connected, which is a desirable property because we assume all of the accounts represented by the nodes in each component belong to a single user. Component A, indicated in Figure 3-5 and enumerated in Table 3.7, is an example of such a component. There are other fully connected components
Figure 3-6: Paired accounts graph properties as a function of threshold $P$. The threshold value 0.782 from equation (3.3) is indicated on the plot.

in Figure 3-5 consisting of more nodes. These components represent users who open many Twitter accounts and retain very similar profile features. Further investigation of these accounts reveals that they are nearly all suspended, suggesting that account suspensions are the driving force behind
the creation of these multiple accounts. As noted earlier, in at least some cases these accounts are created by high-profile online extremists.

Decreasing $P$ from 0.782 appears to rapidly increase the number of false positive classifications. This result becomes quickly apparent in the appearance of a large but loosely connected component in the paired graph structure. For example, reducing the classification threshold to $P = 0.668$ (indicated on the ROC plot in Figure 3-4) increases the profile pairs classified as belonging to the same user to 455. In many cases, these additional pairs appear to be correct classifications. For example, component B from Figure 3-5 appears as a fully connected component using this lower threshold, shown in Figure 3-7. However, we also observe the formation of the less-connected component indicated as “component C” in Figure 3-7. Table 3.9 shows the profile features for the accounts comprising this component, which appear to belong to several different users. The more “loosely connected” components of Figure 3-7 directly correspond to the lower transitivity for this threshold, which is depicted in Figure 3-6.
Table 3.9: Accounts comprising component C.

<table>
<thead>
<tr>
<th>Screen Name</th>
<th>Name</th>
<th>Profile Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAbuAAwlaki</td>
<td>Abu Awlaki</td>
<td>[None]</td>
</tr>
<tr>
<td>abu_alia2</td>
<td>abu alia</td>
<td>[None]</td>
</tr>
<tr>
<td>Abdulllah4510394</td>
<td>Abdulllah</td>
<td>[None]</td>
</tr>
<tr>
<td>abu_abdillah12</td>
<td>Abu Abdullah</td>
<td>[None]</td>
</tr>
<tr>
<td>dwdropz69</td>
<td>Abdulllah</td>
<td>[None]</td>
</tr>
<tr>
<td>Ummabdullaa</td>
<td>Umm Abdullah</td>
<td>[None]</td>
</tr>
<tr>
<td>abouabdullah7</td>
<td>abou abdullah</td>
<td>ff...e0</td>
</tr>
<tr>
<td>AbuAbdullah1400</td>
<td>Abu Abdullah</td>
<td>ff...ff</td>
</tr>
<tr>
<td>abouososama6</td>
<td>Abouososama</td>
<td>[None]</td>
</tr>
<tr>
<td>Abuusamah17</td>
<td>Abu usamah</td>
<td>[None]</td>
</tr>
<tr>
<td>AbuIabulfida</td>
<td>Abu Abdullah</td>
<td>e1...00</td>
</tr>
<tr>
<td>AbuAyman2011</td>
<td>Abu Ayman</td>
<td>[None]</td>
</tr>
<tr>
<td>AbuMuhammad1503</td>
<td>Abu Muhammad</td>
<td>[None]</td>
</tr>
<tr>
<td>abu_malhama4</td>
<td>Abu Malhama</td>
<td>[None]</td>
</tr>
<tr>
<td>moabibkhhab</td>
<td>abu hamad</td>
<td>[None]</td>
</tr>
<tr>
<td>nahida_muhammad</td>
<td>Nahida muhammad</td>
<td>[None]</td>
</tr>
<tr>
<td>abumusab_musab</td>
<td>Abu musab</td>
<td>[None]</td>
</tr>
<tr>
<td>xcon_cp_dc</td>
<td>Abu Musa</td>
<td>[None]</td>
</tr>
<tr>
<td>AbuSaaliha06</td>
<td>Abu Saaliha</td>
<td>[None]</td>
</tr>
<tr>
<td>AbuSaaliha07</td>
<td>Abu Saaliha</td>
<td>00...00</td>
</tr>
<tr>
<td>AbuSaaliha08</td>
<td>Abu Saaliha</td>
<td>00...00</td>
</tr>
<tr>
<td>AbuSaaliha13</td>
<td>Abu Saaliha</td>
<td>00...00</td>
</tr>
<tr>
<td>Abu_swaaliha</td>
<td>abu swaaliha</td>
<td>1e...c3</td>
</tr>
<tr>
<td>Abu_Malhama5</td>
<td>Abu Malhama</td>
<td>bf...00</td>
</tr>
<tr>
<td>omertalhaa</td>
<td>Abu Talha</td>
<td>[None]</td>
</tr>
<tr>
<td>islamobjective</td>
<td>Abu Ramadi</td>
<td>[None]</td>
</tr>
</tbody>
</table>

3.5 Re-following Model

In the previous section we used machine learning to produce a method for efficiently finding groups of accounts that are likely to belong to a single user. In this section, we use the account groups produced from this method in an effort to learn how users tend to reconnect, or re-follow, other user accounts when opening a new account. Being able to predict who users re-follow will allow us to more quickly find these users in the network.

Suppose a user $t$ has his account suspended and decides to open a new account. After getting the account open, $t$ decides to follow some other users. We have observed that in many cases, $t$ will re-follow at least some of the user accounts he was previously following with his suspended account, and it seems reasonable to assume that any suspended user would want to reconnect with some of
the same people he or she was following prior to suspension.

In this section we fit a probability model that assigns a value to each of t’s former friends, giving the probability t will re-follow the former friend upon opening a new Twitter account. We again turn to logistic regression as a means to producing this probability model.

3.5.1 Data

Using the logistic regression model from Section 3.4 with a cutoff of 0.782, we grouped the seed accounts into clusters, each of which we assume belong to the same user. A network representation of the non-singleton clusters is shown in Figure 3-5. Accounts in each cluster were then sorted by account age. After sorting, we compared the friend lists of each pair of consecutive accounts. For each friend of the former account, we created a row in our data set labeled with an indicator of whether or not the same friend was connected to the latter account.

Table 3.10: Example of @MusabGharieb18’s re-following behavior upon opening new account @MusabGharieb13. The entries in the table indicate who these two accounts followed.

<table>
<thead>
<tr>
<th>Friend</th>
<th>@MusabGharieb18</th>
<th>@MusabGharieb13</th>
</tr>
</thead>
<tbody>
<tr>
<td>@poorslave_3</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>@enkorela</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>@StillUkhtMaryam</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>@Yaqub_London</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>

For example, user account 3280844606 (@MusabGharieb18) and user account 3343999888 (@MusabGharieb13) are consecutive accounts belonging to the same user cluster. Table 3.10 shows whether each account was following certain friend accounts. Table 3.11 shows how each of @MusabGharieb18’s friends would then generate a row in the data for this logistic regression model.

Table 3.11: Example data rows resulting from re-following behavior given in Table 3.10. Features are omitted but include, for example, characteristics from each friend’s profile.

<table>
<thead>
<tr>
<th>Friend</th>
<th>[Features]</th>
<th>re-followed (Response)</th>
</tr>
</thead>
<tbody>
<tr>
<td>@poorslave_3</td>
<td>⋯</td>
<td>1</td>
</tr>
<tr>
<td>@enkorela</td>
<td>⋯</td>
<td>-1</td>
</tr>
<tr>
<td>@StillUkhtMaryam</td>
<td>⋯</td>
<td>1</td>
</tr>
<tr>
<td>@Yaqub_London</td>
<td>⋯</td>
<td>-1</td>
</tr>
</tbody>
</table>
3.5.2 Features

In order to obtain a good fit, we included features from the suspended user’s earlier account (e.g., @MusabGharieb18) as well as features from the friend account (e.g., @poorslave). For a suspended user account User0 that was following account Friend, we construct a variety of features which can be broken down into different categories. One set of features deals with the features of the individual accounts of User0 and Friend. A related set of features are about the similarity of the two accounts. There is a category of features that deals with the interactions between the two accounts. Finally, there is a category of features that describe aggregate properties of the neighbors of User0. The complete list of features used in the re-follow model is listed below.

- Friend’s number of Twitter friends (Log).
- Friend’s number of Twitter followers (Log).
- Friend’s number of Tweets (Log).
- Account age difference between Friend and User0.
- Binary indicator of whether Friend was following User0.
- Number of times User0 mentioned Friend in a tweet (Log).
- Number of times User0 retweeted one of Friend’s tweets (Log).
- Number of times User0 replied to one of Friend’s tweets (Log).
- User0’s number of Twitter friends (Log).
- User0’s number of Twitter followers (Log).
- User0’s number of Tweets (Log).
- User0’s number of favorite tweets (Log).
- User0’s total number of retweets (Log).
- Average number of friends of User0’s friends (Log).
- Median number of friends of User0’s friends (Log).
- Standard deviation of the number of friends of User0’s friends (Log).
• Average number of followers of User0’s friends (Log).

• Median number of followers of User0’s friends (Log).

• Standard deviation of the number of followers of User0’s friends (Log).

• Average number of tweets of User0’s friends (Log).

• Median number of tweets of User0’s friends (Log).

• Standard deviation of the number of tweets of User0’s friends (Log).

• Average number of favorite tweets of User0’s friends (Log).

• Median number of favorite tweets of User0’s friends (Log).

• Standard deviation of the number of favorite tweets of User0’s friends (Log).

• Binary indicator of whether Friend’s account authenticity had been verified by Twitter.

• Fraction of User0’s friends that had account authenticity verified by Twitter.

• Binary indicator of whether Friend and User0 had the same account language setting.

3.5.3 Kernel Logistic Regression

Intuitively, some interactions among our set of features might be more predictive than the features themselves. For example, the average number of User0’s friends might not be very useful in estimating the probability User0 re-follows a specific Friend account. However, this value multiplied by Friend’s number of Twitter friends could be very useful. For this reason, we use a quadratic kernel in this logistic regression model, which ensures the regression is fit on all linear and quadratic terms, including pairwise interactions:

\[ K(x, y) = (1 + x^T y)^2 \]

Given a training data set \( \{x_1, x_2, \ldots, x_N\} \), the corresponding logistic regression model is

\[ \hat{p}(x) = \left( 1 + e^{\sum_{i=1}^{N} \alpha_i K(x, x_i)} \right)^{-1}. \]
The parameters $\alpha = (\alpha_1, \ldots, \alpha_N)$ are fit on the training data using an $L_2$-regularized log loss:

$$\alpha = \arg \min_\alpha \sum_{i=1}^{N} \log(1 + e^{-y_i \sum_{i'=1}^{N} \alpha_i K(x_i, x_i')}) + \lambda \alpha^T \alpha,$$

where $y_i$ is the response in the $i$th row of the training data. These responses take value -1 if the Friend was not re-followed, or 1 if the Friend was re-followed, as annotated in Table 3.11. The parameter $\lambda$ serves as the regularization coefficient.

### 3.5.4 Performance

In order to fit this model we used gradient-based optimization methods available in Python’s scipy package [76]. We first selected training (50%), validation (25%), and test (25%) sets randomly from all of the rows of the data and normalized the entire data set based on the values in the training data. Through validation we found that $\lambda = 10^{-5}$ provided the highest AUC. Performance on out-of-sample test data is depicted in Figure 3-8.

![ROC curve for $L_2$-regularized quadratic kernel logistic regression model for predicting re-follows evaluated on out-of-sample test data. (left) Test data and training and validation data can contain the same user. (right) Test data and training and validation data do not contain the same users.](image)

From the figure it appears that we can predict with some accuracy which former friends a suspended user is likely to reconnect with. It is possible, however, that the model is learning re-following preferences of individual users in the data set. To investigate this possibility, we selected new training, validation, and test sets by randomly selecting different user clusters for each and included all of the rows corresponding to these user clusters in the corresponding set. In other words,
each component depicted in Figure 3-5 was assigned as a whole to either the training, validation, or test data, approximately maintaining the 50%-25%-25% ratios. Unlike the previous data partition, this constraint would ensure that all of the rows in Table 3.11 went to the same set, because they belong to the same user.

Using this new data partition, validation and testing were completed on data consisting of entirely different users than those that provided the training data. Through validation we found that $\lambda = 10^{-4}$ provided the highest AUC on this new data partition. Out-of-sample performance suffered, as can be seen in the ROC plot in Figure 3-8. Comparing the performance on each partition provides some interesting insights. First, the AUC for the new partition in Figure 3-8 is 0.66, which indicates that there is some underlying re-following behavior common to the users in our data. However, our ability to predict whether or not a suspended user will re-follow an old friend increases substantially when we include that user’s past behavior in the training data. The difference in performance gives us an idea of how useful it is to have data on a specific user’s past behavior when predicting whom the user will re-follow.

Because we used a quadratic kernel logistic regression, the expressions for the fit models are not easy to interpret. However, their performance shows that we can predict with some accuracy the re-following behavior of a suspended user, even in the absence of previous re-following behavior, based solely on the re-following behavior of others. We make use of this capability in the next section, where we develop a method to search for a suspended user’s new account.

### 3.6 Suspended User Search

We now make use of our findings from the previous sections to address another relevant problem. We have observed multiple incidences in our data of suspended users quickly creating a new Twitter account in order to continue their unethical activity, as exemplified in Table 3.3. In these instances it would be useful for those tasked with monitoring nefarious users, such as social media service providers or intelligence community personnel, to find an efficient way to search for the suspended user’s new account.

We assume we are given a target user whose account has been suspended by Twitter. We have stored the target user’s account information, including lists of the target’s friends and followers. From this information we wish to locate the target user’s new Twitter account, if one exists, as efficiently as possible. Our approach to solving this problem is to query the followers of each of the
target user’s known Twitter “friend” accounts, prior to suspension, and search the results for a new account belonging to the target user.

Our network search model is a generalization of the model presented in Chapter 2 and illustrated in Figure 2-1. We can think of each of the target’s former friends $i$ as an urn containing $N_i$ marbles, which represent the neighbors of $i$. If the target has connected to former friend $i$, then he is among $i$’s neighbors and a single red marble is one of the $N_i$ marbles in urn $i$. Excepting these red marbles, all marbles in all urns are blue.

Each follower query can be thought of as choosing a nonempty urn $j$ in the multi-urn model and removing some fixed number of its marbles. The number of marbles removed is determined by the query method used and, unlike our network search in Chapter 2, can be more than one. Having a red marble among those removed represents finding the target user’s account, and the search terminates. In this analogy, checking if a marble is red means matching the queried account with the target user’s suspended account, which can be done using the method presented in Section 3.4.

### 3.6.1 Suspended User Search Model

Let $\mathcal{V}$ be the set of known friend accounts. These are the accounts that the target user was following prior to begin suspended. For each known friend $i \in \mathcal{V}$, let $N_i$ be the number of Twitter accounts that are following $i$. These quantities are easily obtained through the Twitter API.

Using the Twitter API it is possible to obtain a list of the followers of a specified user, provided the user has not enabled privacy protection on the account. Twitter offers two methods for executing these queries: GET followers/list and GET followers/ids. Both methods are rate limited to 15 queries within any 15-minute time period. GET followers/list returns standard Twitter user profile information for each follower, but only returns up to 200 profiles per query. GET followers/ids has the same rate limit, but returns up to 5,000 user IDs per query [138].

Each method can be cursoried so that subsequent queries of the same user continue to produce unique results, until all of the user’s followers’ profiles or IDs have been obtained. For our analysis, we set $N_M$ as the maximum number of unique followers obtained per query, although in practice we assume this number to be 5,000 as established in the GET followers/ids method. Therefore if we have queried user $i$’s followers $n$ times, we expect the next query of user $i$’s followers to return $\min\{N_M, N_i - nN_M\}$ new results, provided $i$ still has unqueried followers ($N_i - nN_M > 0$).

Additionally, we make the following assumptions:
1. After being suspended, the target user creates a new account with probability $\rho_0$, which we refer to as the a priori existence probability. If the target has not created a new account, then he does not have a node in the network and will not be found through follower queries. The value of $\rho_0$ quantifies the searcher’s belief that the target exists in the network.

2. If the target user creates a new account, he reconnects with each former friend $i \in \mathcal{V}$ with some probability $\varphi_i$, which can be estimated from previous account data as was done in Section 3.5. We refer to this as the reconnection probability to former friend $i$.

3. Reconnections to former friends are independent; whether or not the search target reconnects with former friend $i$ does not affect the probability he reconnects with former friend $j \neq i$.

4. If the target user is following user $i \in \mathcal{V}$, then each account returned in each query of $i$’s followers is equally likely to be the target’s account.

5. The searcher can quickly and accurately determine whether an account obtained from a follower query is the target user’s account. This can be done using the approach developed in Section 3.4.

The search process is modeled as the execution of follower queries in discrete stages. In each stage $t \in \{0, 1, \ldots, N-1\}$, the searcher chooses one of the target user’s former friend accounts and executes a follower query. Here, $N$ is the total number of queries required to examine all of the followers of all former friends, and is assumed to be finite. If the target user’s new account is among the query results, the search terminates. Otherwise, the searcher executes another query unless all $N$ queries have been exhausted or the searcher concludes that the target has not created a new account.

The objective of the search is to minimize the total number of queries. For consistency with the development in Chapter 2, we do not consider the cost of a query that succeeds in returning the target user’s new account. Therefore, the objective in our search model is to minimize the number of unsuccessful search queries. The best result possible would be to find the search target in the first query, in which case there are zero unsuccessful queries. Because of the stochastic nature of this process, we say that a search policy is optimal if it minimizes the expected number of unsuccessful queries.
3.6.2 Initialization

We assume that data collected on the target user provides a list of known former friend accounts. Using the Twitter API, it is relatively easy to determine which of these accounts are still active, whether or not they are “private,” and their follower counts. We initialize set $\mathcal{V}$ as the set of all former friend accounts that are active at the time of search execution, that have followers that can be queried (i.e., have a positive number of followers and are not “private” accounts). We use the follower counts for these accounts to initialize $N_i, i \in \mathcal{V}$.

This search model also requires an initial probability that the target user would reconnect with each former friend $i \in \mathcal{V}$, given he has created a new account. Let $A$ be the event that the target has created a new account, $B_i$ be the event that the target is following former friend $i \in \mathcal{V}$, and $B = \bigcup_{i \in \mathcal{V}} B_i$ be the event that the target has reconnected with at least one former friend. From our definitions above, we can write $\varphi_i = \mathbb{P}(B_i|A)$. We can obtain the value of this probability using the approach presented in Section 3.5. Note that event $B$ can also be interpreted as the event we can find the target user by exhaustively querying the followers of all former friends. Using our independence assumption we have

$$\mathbb{P}(B^c|A) = 1 - \mathbb{P}(B|A) = \prod_{i \in \mathcal{V}}(1 - \varphi_i).$$

We also must select a value for the a priori existence probability $\rho_0$, which can be done based on the beliefs of experts in the relevant domain. As the search process progresses, the conditional existence probability will evolve. The search terminates if the target user is found or all follower queries are exhausted. In addition to these criteria, a searcher might want to terminate the search upon achieving reasonable certainty that the target user has not created a new account. We allow for this termination criterion by including a termination conditional existence probability $\bar{\rho}$. If at any stage the conditional existence probability falls below $\bar{\rho}$, the search terminates and the searcher concludes that the target has not created an identifiable new account. If $\bar{\rho}$ is set to zero, then the search continues until the target is found or until all follower queries are exhausted.

3.6.3 The Discrete Stochastic Search Process

As we have suggested, the search process can be modeled as a set of urns, each representing a former friend. Urn $i \in \mathcal{V}$ has $N_i$ marbles, which represent former friend $i$’s followers. In each stage the searcher chooses a former friend (or urn) and executes a follower query, receiving up to $N_M$ results.
(or drawing at most \(N_M\) marbles from the urn). The search continues until one of the following occurs:

- The target’s new account is found (a red marble is among those drawn),
- The probability the target has created a new account falls below the termination probability \(\bar{\rho}\)
- The queries of former friends’ followers are exhausted (there are no marbles left in any of the urns).

**Policy**

Suppose we consider a valid policy as any sequence of former friend queries in which each former friend is exactly exhaustively queried. In other words, if we let \(u = (u_0, u_1, \ldots, u_{N-1})\) be a policy in which former friend \(u_t \in \mathcal{V}\) is queried in stage \(t\), then \(u\) is valid if and only if

\[
|x_i(t)| = \left\lceil \frac{N_i}{N_M} \right\rceil \quad \forall i \in \mathcal{V}.
\]

Notice that any valid policy can be completely specified in advance as an ordering of follower queries that is executed until one of the three termination criteria are met. As long as the target is not found, state transitions are deterministic and can be enumerated a priori. Except for the decision to terminate, there is no benefit to making policy decisions during the search. Unsuccessful search results do not provide any additional insight into which ordering of queries might yield a lower cost.

**System State and Transitions**

In order to analyze the dynamics of the system we define the system state, \(x(t)\), at stage \(t\) as either a \(|\mathcal{V}|\)-dimensional vector in which the \(i\)th element \(x_i(t)\) is the number of follower queries that have been executed on former friend \(i \in \mathcal{V}\) in previous stages, or a terminal state, “Terminate.” At stage \(t = 0\), no queries have been executed and presumably the search has not terminated, so that \(x(0) = 0\). In any non-terminal state, let the vector \(x(t)\) be specified as a function of the policy being executed:

\[
x_i(t) = |\{\ell < t : u_\ell = i\}| \quad \forall i \in \mathcal{V}.
\]
State transitions in this system are a function of the current state, the policy, and a stochastic input representing whether the target account is found as a result of the current stage query. Let

\[ w(x(t), i) = \begin{cases} 
0, & \text{Target is not found querying } i \text{ from state } x(t) \\
1, & \text{Target is found querying } i \text{ from state } x(t). 
\end{cases} \]

We now have all of the definitions needed to write the state transition function that governs this search model.

\[ x(t + 1) = f(x(t), u_t, w(x(t), u_t)) \]

\[ = \begin{cases} 
\text{"Terminate,"} & w(x(t), u_t) = 1 \text{ or other termination criterion are met} \\
x(t) + e_{u_t}, & \text{otherwise.} 
\end{cases} \]

Here, \( e_i \) represents the \( i \)-th unit vector.

### 3.6.4 Search Process Dynamics

We define the function

\[ \psi_i(u, t) = \max \left\{ \frac{x_i(t)N_M}{N_i}, 1 \right\} \]

\[ = \begin{cases} 
\frac{x_i(t)N_M}{N_i}, & x_i(t) = 0, 1, \ldots, \left\lceil \frac{N_i}{N_M} \right\rceil - 1 \\
1, & x_i(t) = \left\lceil \frac{N_i}{N_M} \right\rceil 
\end{cases} \]

as the fraction of former friend \( i \)'s followers that have been queried before stage \( t \) when executing valid policy \( u \) (or, using the urn analogy, the fraction of marbles that have been removed from urn \( i \) at stage \( t \)), conditioned on not having found the target user prior to stage \( t \). This function captures the assumption that, provided former friend \( i \) has more than \( N_M \) unqueried followers remaining in stage \( t \), the query returns \( N_M \) followers. If former friend \( i \) has fewer than \( N_M \) unqueried followers remaining in stage \( t \), then the query will return all of the remaining unqueried followers.

This function is strictly increasing at a constant rate of \( \frac{N_M}{N_i} \) as \( x_i(t) \) increases from 0 to \( \left\lceil \frac{N_i}{N_M} \right\rceil - 1 \). It continues to increase, at a possibly slower rate, in the final or \( \left( \left\lceil \frac{N_i}{N_M} \right\rceil \right) \)th query of former friend \( i \). Because \( x_i(t) \) is nondecreasing in \( t \), we can conclude that \( \psi_i(u, t) \) is also nondecreasing in \( t \).

For example, suppose a certain former friend has 12,000 followers and that each follower query
returns at most $N_M = 5,000$ followers. Then, the first and second query of this former friend will return 5,000 followers each, while the final query will only return 2,000 followers. In general, we expect the first $\left\lceil \frac{N_i}{N_M} \right\rceil - 1$ follower queries of former friend $i \in \mathcal{V}$ to return $N_M$ results, while the final query returns $N_i - \left(\left\lceil \frac{N_i}{N_M} \right\rceil - 1\right) N_M$ results. This irregularity results in final queries of former friends affecting the system dynamics differently than the preceding queries of the same former friends.

**Conditional Existence Probability**

We now develop an expression for the conditional existence probability, i.e., the probability that the target user has created a new account conditioned on having reached a certain non-terminal state, $x(t)$. Let $A$ be the event that the target user has created a new account. For simplicity of notation, we condition directly on the state vector $x(t)$ to denote the event that this state has been reached without finding a target user’s new account, so that $\rho(t) = P(A|x(u, x(t)))$ is the new account existence probability conditioned on having reached state $x(t)$ when executing valid search policy $u$ without having found the target account. Note that $\rho(0) = \rho_0$, the initialization value.

Using Bayes’ rule, the conditional existence probability is

$$\rho(t) = \rho_0 \left( \frac{\prod_{i \in \mathcal{V}} (1 - \psi_i(u, t)\varphi_i)}{1 - \rho_0 + \rho_0 \prod_{i \in \mathcal{V}} (1 - \psi_i(u, t)\varphi_i)} \right).$$

The terms inside the products are the probabilities of not finding the target account among the followers of each former friend $i$, given that $\psi_i(u, t)$ fraction of those followers have been queried and examined. Multiplying these probabilities together implicitly relies on our assumption that the target user reconnects to his former friends independently.

The expression for $\rho(t)$ is the initial existence probability multiplied by a ratio of two linear functions of the product $\prod_{i \in \mathcal{V}} (1 - \psi_i(u, t)\varphi_i)$. Because $\psi_i(u, t) \leq 1 \forall i \in \mathcal{V}$ and is nondecreasing in $t$, $\prod_{i \in \mathcal{V}} (1 - \psi_i(u, t)\varphi_i)$ is nonincreasing in $t$. The coefficient in the denominator ($\rho_0$) is no more than that of the numerator (1), and therefore the conditional existence probability is nonincreasing in $t$ and converges to 0 as $\prod_{i \in \mathcal{V}} (1 - \psi_i(u, t)\varphi_i)$ decreases to 0. This monotonicity property aligns with intuition: the more the social network is searched without finding the target user, the less likely it is that the target user exists in the network.

Other than the conditional existence probability at each stage, the value of the initial existence probability $\rho_0$ does not affect the system dynamics. Implicit in the execution of the search is the
assumption that the search target has created a new account and reconnected to former friends in a way that can be represented by a probability model. The utility of including an existence probability in the model is that it enables the searcher set a search termination criterion when he is sufficiently convinced that the target user has not created a new account, based on the value of the conditional existence probability.

**Conditional Reconnection Probabilities**

The conditional probability that the target user has reconnected with former friend \(i\), given he has created a new account and has not been found by stage \(t\) when applying search policy \(u\), can also be calculated using Bayes’ rule. Recall that \(A\) is the event that the target user created a new account and \(B_i\) is the event that the target user has reconnected with friend \(i\). Then we have

\[
P(B_i|A, u, w(x(t), i)) = 0 = \frac{P(w(x(t), i) = 0|B_i, u, A)P(B_i|u, A)}{P(w(x(t), i) = 0|B_i, u, A)P(B_i|u, A) + (1 - P(B_i|u, A))} \\
= \varphi_i \left( \frac{1 - \psi_i(u, t)}{1 - \psi_i(u, t) \varphi_i} \right) \\
= \begin{cases} 
\varphi_i \left( \frac{N_i - x_i(t)N_M}{N_i - x_i(t)N_M} \right), & x_i(t) = 0, 1, \ldots, \left(\left\lceil \frac{N_i}{N_M} \right\rceil - 1\right) \\
0, & x_i(t) = \left\lceil \frac{N_i}{N_M} \right\rceil 
\end{cases}
\]

Observe that this probability is the original probability multiplied by the ratio of two linear functions of \(x_i(t)\). Because the numerator decreases at a faster rate than the denominator, this probability is strictly decreasing as \(x_i(t)\) increases from 0 to \(\left\lceil \frac{N_i}{N_M} \right\rceil\), provided \(\varphi_i > 0\). Just as with the conditional existence probability, the monotonicity of this conditional probability matches intuition: the more we query the followers of a certain former friend without finding the target, the less likely it becomes that the target has reconnected with this former friend.

**Distribution of** \(w(x(t), i)\)

The probability of finding the target when querying former friend \(i \in V\) from state \(x(t)\) is found using the multiplication rule. Note that the event

\[
\{w(x(t), i) = 1\} \subseteq B_i \subseteq A.
\]
Therefore,

\[ P(w(x(t), i) = 1) = P(w(x(t), i) = 1|B, A, x(t))P(B_i|A, x(t))P(A|x(t)) \]

\[ = \begin{cases} \varphi_i \left( \frac{N_M}{N_i - \varphi_i x_i(t) N_M} \right) & x_i(t) = 0, 1, \ldots, \left\lfloor \frac{N_i}{N_M} \right\rfloor - 2 \\ \varphi_i \left( \frac{N_i - x_i(t) N_M}{N_i - \varphi_i x_i(t) N_M} \right) & x_i(t) = \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \end{cases} \]  \tag{3.5}

\[ P(w(x(t), i) = 0) = \begin{cases} \varphi_i \left( \frac{N_M}{N_i - \varphi_i x_i(t) (t+1) N_M} \right) & x_i(t) = 0, 1, \ldots, \left\lfloor \frac{N_i}{N_M} \right\rfloor - 2 \\ \frac{(1-\varphi_i) N_i}{N_i - \varphi_i x_i(t) N_M} & x_i(t) = \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \end{cases} \]  \tag{3.6}

This expression offers several important insights into the dynamics of this search model. First note that conditioned on the existence of a new target account,

\[ P(w(x(t), i) = 1|A, x(t)) = \begin{cases} \varphi_i \left( \frac{N_M}{N_i - \varphi_i x_i(t) N_M} \right) & x_i(t) = 0, 1, \ldots, \left\lfloor \frac{N_i}{N_M} \right\rfloor - 2 \\ \varphi_i \left( \frac{N_i - x_i(t) N_M}{N_i - \varphi_i x_i(t) N_M} \right) & x_i(t) = \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \end{cases} \]  \tag{3.5}

\[ P(w(x(t), i) = 0|A, x(t)) = \begin{cases} \varphi_i \left( \frac{N_M}{N_i - \varphi_i x_i(t) (t+1) N_M} \right) & x_i(t) = 0, 1, \ldots, \left\lfloor \frac{N_i}{N_M} \right\rfloor - 2 \\ \frac{(1-\varphi_i) N_i}{N_i - \varphi_i x_i(t) N_M} & x_i(t) = \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \end{cases} \]  \tag{3.6}

We refer to equation \(3.5\) as the probability of success when querying former friend \(i\) from state \(x(t)\). Likewise, equation \(3.6\) is the failure probability when querying former friend \(i\) from state \(x(t)\). Given the target user has created a new account, the success probability for a specific friend \(i \in \mathcal{V}\) is strictly increasing as \(x_i(t)\) increases from 0 to \(\left\lfloor \frac{N_i}{N_M} \right\rfloor - 2\), and is therefore nondecreasing over the corresponding stages \(t\). However, this monotonicity property does not always hold for the final query. As we have discussed, the final query of \(i\) does not necessarily return the same number \((N_M)\) of results as previous queries of \(i\), and has a different functional form for probability of success given in equation \(3.5\).

Figure 3-9 illustrates this monotonicity property for two initial conditions. In both of the plotted trajectories, \(\left\lfloor \frac{N_i}{N_M} \right\rfloor = 20\). For former friend 1, \(N_1 \mod N_M = 0\) and all queries return the same number \((N_M)\) of results. In this case the probability of finding the target is strictly increasing over all queries of this former friend’s followers. The second former friend’s success probabilities depicted in Figure 3-9 do not have this characteristic, and the final query returns fewer results than the previous 19 queries. In this case, we observe that the probability of finding the target is strictly increasing over the first 19 queries, but decreases in the final query because this query returns fewer results.

This monotonicity property is an extension of Theorem 7. As a final note on this property, we
observe that this result holds even if we remove the conditioning on $A$. If in stage $t$ the searcher queried the followers of former friend $i$ and did not find the target user, then in stage $t+1$,

$$\mathbb{P}(w(x(t+1), i) = 1) > \mathbb{P}(w(x(t), i) = 1),$$

for all $0 \leq x_i(t+1) < \left\lfloor \frac{N_i}{M} \right\rfloor - 1$, $\varphi_i > 0$, and $\rho(t) > 0$.

### 3.6.5 Analysis: $\bar{\rho} = 0$

We provide analysis for the case in which we initialize $\bar{\rho} = 0$, i.e., we continue to search until either the target account is found or all follower queries have been exhausted. If we were searching for a suspended user’s new account, one course of action would be to first execute the query that was most likely to reveal the account. However, as we demonstrated in Chapter 2, this approach does not always yield the optimal policy. In this section we provide a characterization of the optimal policy that naturally extends from the optimality condition derived for independent urns.
Expression for Expected Policy Cost

We now derive an expression for policy cost when $\bar{\rho} = 0$. Let $u$ be a valid police and $C_u$ be the number of unsuccessful queries, or cost of policy $u$. Because $C_u$ can only take nonnegative integral values $0, 1, \ldots, N$,

$$E[C_u] = \sum_{t=0}^{N-1} P(C_u > t)$$
$$= \sum_{t=0}^{N-1} P(C_u > t|A)P(A) + \sum_{t=0}^{N-1} P(C_u > t|A^c)P(A^c)$$
$$= \rho_0 \sum_{t=0}^{N-1} \prod_{k=0}^{t} P(w(x(t), u_t) = 0|A) + N(1 - \rho_0). \quad (3.7)$$

The optimal search policy is the valid policy that minimizes this expression. Formally,

$$u^* = \arg \min_{u \in U} E[C_u]$$
$$= \arg \min_{u \in U} \left\{ \rho_0 \sum_{t=0}^{N-1} \prod_{k=0}^{t} P(w(x(t), u_t) = 0|A) + N(1 - \rho_0) \right\}$$
$$= \arg \min_{u \in U} \sum_{t=0}^{N-1} \prod_{k=0}^{t} P(w(x(t), u_t) = 0|A). \quad (3.8)$$

where $u^*$ is the optimal policy and $U$ is the set of valid policies. Recall from equation (3.4) that the vectors $x(t)$ can be written as a function of the search policy. Not surprisingly, if we commit to exhausting all possible queries in our search for the target, the initial existence probability $\rho_0$ does not affect policy optimality.

In order to simplify notation, we define the probability $q_u(t) = P(w(x(t), u_t) = 0|A, x(t))$. This is the probability of failing to find the target’s new account when executing the $t$th query in policy $u$. This probability is specified in equation (3.6), and allows us to rewrite the objective function in equation (3.8) as

$$u^* = \arg \min_{u \in U} \sum_{t=0}^{N-1} \prod_{k=0}^{t} q_u(t).$$

Optimality Conditions

In Theorem 2, we demonstrated that there exists a block policy, in which each urn $i \in V$ is exhaustively queried before moving on to another urn, that is optimal in any multi-urn search.
problem. We now provide a corollary which generalizes the result in Theorem 2 to allow for a single query to return multiple results.

**Corollary 7. (Necessary Conditions for Optimality)** If in a suspended user search, follower queries of former friends are executed until either the target user is found or all queries have been exhausted, then any optimal policy must satisfy the following conditions:

1. The first \( \left\lceil \frac{N_i}{N_M} \right\rceil - 1 \) queries of each former friend \( i \in \mathcal{V} \) are executed in succession in a single block.

2. For all friends \( i \in \mathcal{V} \) such that

\[
\frac{N_i}{N_M \varphi_i} - \frac{1}{2} \left\lceil \frac{N_i}{N_M} \right\rceil > \frac{(1 - \varphi_i) N_i}{\varphi_i \left( N_i - \left\lceil \frac{N_i}{N_M} \right\rceil N_M + N_M \right)},
\]

\( (3.9) \)

all \( \left\lceil \frac{N_i}{N_M} \right\rceil \) queries of \( i \)'s followers are executed in succession.

The first part of Corollary 7 follows from the monotonicity of the success probability. If querying former friend \( i \) is optimal in stage \( t \), and in the next stage \( (t+1) \) the success probability for \( i \) has increased while success probabilities for all \( j \in \mathcal{V} \setminus i \) have remained the same, then intuitively it would be optimal to query \( i \) again in stage \( t+1 \).

The condition in equation (3.9) is related to how the success probability changes in the final query of each former friend. If

\[
\frac{N_j}{N_M \varphi_j} - \frac{1}{2} \left\lceil \frac{N_j}{N_M} \right\rceil > \frac{(1 - \varphi_j) N_j}{\varphi_j \left( N_j - \left\lceil \frac{N_j}{N_M} \right\rceil N_M + N_M \right)},
\]

then the final query of \( i \) has a lower cost than the previous queries of \( i \). This is the case depicted in Figure 3-9 for former friend 1. In this case, querying all of \( i \)'s followers in succession starting at any stage \( t \) is more valuable than executing only the first \( \left\lceil \frac{N_i}{N_M} \right\rceil - 1 \) queries, and any optimal policy will include all of these queries in a single block.

If on the other hand

\[
\frac{N_j}{N_M \varphi_j} - \frac{1}{2} \left\lceil \frac{N_j}{N_M} \right\rceil < \frac{(1 - \varphi_j) N_j}{\varphi_j \left( N_j - \left\lceil \frac{N_j}{N_M} \right\rceil N_M + N_M \right)},
\]

then the final query of former friend \( i \) has a higher cost than the previous query. This is the case of former friend 2 depicted in Figure 3-9. In this case querying all of \( i \)'s followers in succession starting
at any stage \( t \) is less beneficial, in terms of minimizing cost, than executing only the first \( \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \) queries. The optimal policy might separate the final query of \( i \) from the first \( \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \) queries in this case.

If the inequality in equation (3.9) is instead satisfied with equality, then executing all of the queries of \( i \)'s followers in succession from any stage \( t \) essentially provides the same benefit as executing only the first \( \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \) queries. In this case, an optimal policy will always exist in which these queries are executed together in a single block, but alternative policies with equal cost might also exist in which the first \( \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1 \) queries of \( i \) are separated from the final query.

Like Theorem 2, Corollary 7 establishes that the optimal policy is a block policy, but it does not specify the details of this policy. We now provide a generalization of Theorem 4, which provides a full characterization of an optimal policy.

**Corollary 8. (Necessary and Sufficient Conditions for Optimality)** In a suspended user search, define

\[
\gamma(x(t), i) = \begin{cases} 
\frac{1}{\varphi_i} \left[ \frac{N_i}{N_M \varphi_i} - \frac{N_M}{2N_i} \left( \frac{N_i}{N_M} \right) - 1 \right], & x_i(t) = 0, 1, \ldots, \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1; \\
\frac{\varphi_i}{N_i - \frac{N_i}{N_M} N_M + N_M}, & x_i(t) = \left\lfloor \frac{N_i}{N_M} \right\rfloor - 2; \\
\frac{\varphi_i}{N_i - \frac{N_i}{N_M} N_M + N_M}, & x_i(t) = \left\lfloor \frac{N_i}{N_M} \right\rfloor - 1; \\
\infty, & \text{otherwise}.
\end{cases}
\]

A valid policy is optimal if and only if it satisfies the conditions in Corollary 7 and it minimizes \( \gamma(x(t), i) \) in each stage, i.e.,

\[
u_t = \arg\min_{i \in \mathcal{V}} \gamma(x(t), i) \quad t = 0, 1, \ldots, N - 1.
\]

The function \( \gamma(x(t), i) \) follows from the expression for optimality in the proof of Theorem 4 when comparing the costs of policies which swap the order of querying former friend \( i \) with another former
friend. Corollary 8 simply says that always choosing the former friend that minimizes $\gamma(x(\tau), i)$ produces an optimal policy. The different cases for $\gamma(x(\tau), i)$ correspond to different remaining followers to query of the former friends along with the optimality conditions from Corollary 7.

The first case corresponds to the condition in equation 3.9. As discussed, this condition implies that executing all queries of $i$ in a single block is more beneficial than executing only the first $\left\lceil \frac{N_i}{N_M} \right\rceil - 1$ queries. The other cases follow similar logic: the second case is the value function for the first $\left\lceil \frac{N_i}{N_M} \right\rceil - 1$ queries of former friend $i$, and the condition indicates that executing only these queries in a single block is best. The third case is for the final query of former friend $i$, and the fourth condition sets $\gamma(x(t), i)$ to infinity if there are no queries remaining for $i$.

### 3.6.6 Results

Using the classification results from Section 3.4, we identified 169 account pairs from our ISIS seed users for testing. Each pair of accounts consisted of an earlier account, which had been suspended, and an account opened later that belonged to the same user. Without being able to verify exactly when account suspensions took place, we assumed the later account in each case was opened or used in response to the former account’s suspension. Having collected the friends and followers lists for all of these accounts, we were able to evaluate the performance and effectiveness of the search policy we developed.

<table>
<thead>
<tr>
<th>Pair</th>
<th>Former friends</th>
<th>Reconnection %</th>
<th>Max Queries $(N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>40.00%</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>310</td>
<td>59.68%</td>
<td>6609</td>
</tr>
<tr>
<td>3</td>
<td>94</td>
<td>17.02%</td>
<td>247</td>
</tr>
<tr>
<td>4</td>
<td>87</td>
<td>21.84%</td>
<td>431</td>
</tr>
<tr>
<td>5</td>
<td>185</td>
<td>8.11%</td>
<td>198</td>
</tr>
<tr>
<td>6</td>
<td>84</td>
<td>22.62%</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>63</td>
<td>9.52%</td>
<td>12007</td>
</tr>
<tr>
<td>8</td>
<td>189</td>
<td>4.23%</td>
<td>2078</td>
</tr>
<tr>
<td>9</td>
<td>257</td>
<td>88.72%</td>
<td>4312</td>
</tr>
<tr>
<td>10</td>
<td>109</td>
<td>30.28%</td>
<td>152</td>
</tr>
<tr>
<td>11</td>
<td>302</td>
<td>82.12%</td>
<td>5559</td>
</tr>
<tr>
<td>12</td>
<td>344</td>
<td>22.67%</td>
<td>1314</td>
</tr>
<tr>
<td>13</td>
<td>181</td>
<td>9.94%</td>
<td>190</td>
</tr>
<tr>
<td>14</td>
<td>87</td>
<td>3.45%</td>
<td>2965</td>
</tr>
<tr>
<td>15</td>
<td>221</td>
<td>2.26%</td>
<td>2654</td>
</tr>
</tbody>
</table>
From the set of 169 account pairs, we randomly chose 15 for testing. Table 3.12 shows the number of former friends, the reconnection rate, and the total number of queries possible (or policy length) for each of these account pairs. For each account pair, we identified the friends from the earlier (suspended) account as the “former friends” of the subsequent account. For each of these former friends we determined their reconnection probability using the logistic regression classifier from Section 3.5. We also had the number of followers for each former friend stored in our data set. We assumed that all of the former friend accounts were still active when the second account was opened. Finally, we initialized \( \rho_0 = 1 \). This initial value is useful because it reduces the expression for expected policy cost to the objective function in equation (3.8) and allows for direct comparison of actual performance with our theoretical expected number of unsuccessful queries.

In order to evaluate policy performance, we consider the following policies:

- **Optimal.** This is a policy that minimizes expected cost, found using the necessary and sufficient conditions in Corollary 8.

- **Greedy.** This policy maximizes the probability of finding the new account at each stage. Because this probability strictly increases for each former friend \( i \in \mathcal{V} \) every time \( i \) is queried, excepting the final query of \( i \), this policy always meets the necessary condition for optimality given in Corollary 7.

- **Min-\( \mathcal{N} \).** This policy selects the former friend with the minimum number of unqueried followers at each stage. Because these values strictly decrease for each former friend \( i \in \mathcal{V} \) with each query of \( i \), this policy always meets the necessary condition for optimality given in Corollary 7.

- **Max-\( \mathcal{P} \).** This policy selects the former friend with the highest conditional reconnection probability at each stage. Because conditional reconnection probabilities strictly decrease for each former friend \( i \in \mathcal{V} \) with each query of \( i \), this policy does not necessarily meet the conditions in Corollary 8.

- **Random.** This policy randomly chooses a query from those that are possible at each stage.

**Comparison of Expected Costs**

We computed the expected cost for each policy using equation (3.7). These values do not account for our knowledge of the true reconnections of the second account in each case. Instead, we assume
that our probability model is correct in these computations.

Table 3.13: Cost comparisons for different policies.

<table>
<thead>
<tr>
<th>Pair</th>
<th>Expected Cost</th>
<th></th>
<th></th>
<th>Actual Cost</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimal Greedy Min-N Max-P Random</td>
<td>Optimal Greedy Min-N Max-P Random</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5.72 5.74 9.18 5.89 7.93</td>
<td>3.00 3.00 2.00 2.74 1.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.26 2.27 4.15 88.23 68.87</td>
<td>0.00 0.00 2.00 44.16 20.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.22 1.22 2.00 6.09 4.91</td>
<td>1.00 1.00 6.00 6.28 11.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.20 1.20 2.74 20.48 8.90</td>
<td>2.00 2.00 15.00 26.81 20.79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2.96 2.96 9.27 3.36 6.19</td>
<td>5.00 5.00 15.00 6.56 10.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.96 0.96 2.52 4.43 1.86</td>
<td>1.00 1.00 7.00 5.53 4.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>103.51 103.98 107.53 400.48 2170.52</td>
<td>5.00 5.00 12.00 283.13 1582.28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4.98 5.10 9.05 74.36 71.86</td>
<td>6.00 6.00 136.00 82.50 242.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2.28 2.28 4.73 80.68 57.27</td>
<td>0.00 0.00 2.00 54.75 5.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.01 1.01 2.99 8.64 2.27</td>
<td>3.00 3.00 3.00 13.76 3.27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.89 0.89 2.02 126.65 38.17</td>
<td>0.00 0.00 0.00 111.03 18.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2.88 2.88 6.98 42.15 18.78</td>
<td>0.00 0.00 6.00 44.57 19.57</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1.50 1.50 3.82 3.26 2.52</td>
<td>1.00 1.00 10.00 3.41 9.57</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>8.84 8.85 15.28 141.16 322.96</td>
<td>4.00 4.00 52.00 150.62 736.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.17 1.17 2.84 61.06 20.02</td>
<td>7.00 7.00 61.00 143.00 390.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-10: Box plots of the percentage above optimal of the (left) expected cost and (right) actual cost of different search polices for ISIS users in Twitter.

Table 3.13 gives the expected costs computed for each policy. Expected cost values are analytically computed in all cases except for the random policy. In order to estimate expected cost for a random policy, we generated 500 random policies and computed the expected cost for each. The average of these 500 expected costs is reported as the random policy expected cost in Table 3.13. For a more visual presentation of our results, we also show in Figure 3-10 box plots of the percentage by which each policy exceeds the optimal policy for the different users.

The results show that in many cases, the greedy policy and the optimal policy achieve the same cost. Comparison of these two policies reveals that they are very similar in all cases. The Min-N
policy also produces costs close to those of the optimal and greedy policies, while the Max-$P$ and random policies have a substantially higher costs in many cases.

**Comparison of Actual Costs**

In this section we compare the performance of the different policies in finding the target user based on the actual reconnections. If the target users tended to reconnect in accordance with our probability model we would expect these actual cost values to be similar to the expected costs in Table 3.13. In cases in which the target user reconnected to former friends in a way that would be very unlikely according to our probability model, the actual policy costs might differ substantially from the expected costs. The actual costs for each policy are reported in Table 3.13 with the corresponding box plots in Figure 3-10. The values reported are the expected number of queries one would have to execute before finding the target user, conditional on the target user’s actual reconnections.

In some of the 15 cases, the actual costs in Table 3.13 differ substantially from the expected costs. However, the same trend holds: the optimal and greedy policies tend to perform the best, and are nearly indistinguishable in terms of costs. The Min-$N$ policy performs as well or nearly as well as the optimal policy in some cases, but in a few cases it is much worse. The Max-$P$ and random policies tend to perform poorly, especially when the target user has not connected with very many former friends (see Table 3.12). Using the optimal or greedy policies can result in substantial cost savings in these cases.

Account pair 1 provides an example of a case where a random policy can outperform the optimal policy in practice. The reconnection rate for this target user was 40% (from Table 3.12), but the target did not reconnect with the most probable former friends, according to our probability model (in actuality, it is possible these accounts were suspended when the target opened the new account). From Table 3.12, it is apparent that most of the 35 former friends have fewer than 5,000 followers, because the valid policy length is at most 38 queries. The random policy performs approximately as we would expect in this case: each random query has approximately a 40% chance of returning the target. From the well-known geometric probability distribution, the expected number of failures before the first success is 1.5, which is very close the value reported in Table 3.13.

In each of the 15 account pairs, the optimal, greedy and Min-$N$ policies located the target user when querying a former friend that had fewer than 5,000 followers. For this reason, the actual number of queries in these cases was deterministic, resulting in the integer costs reported in the
Table 3.13. For pair 15, for example, the optimal policy would always find the target user on the 8th query because this is the first former friend in the policy to whom the target user had reconnected, and a single query retrieves all of this former friend’s followers. In this application, many of the former friend accounts have fewer than 5,000 followers and are therefore exhausted in a single followers query. These accounts, when coupled with a high reconnection probability, are very valuable in a search policy. Both the greedy and the optimal policies prioritize queries of former friends with relatively high reconnection probabilities and low numbers of followers.

3.6.7 Discussion: \( \bar{\rho} > 0 \)

When an existence probability threshold is applied as a termination criterion, Corollaries 7 and 8 no longer hold. However, the queries that have the highest probability of finding the target user are also the queries that have the largest effect on reducing the conditional existence probability. We conjecture that the optimal policy in the case for which \( \bar{\rho} > 0 \) will be the same as the optimal policy when \( \bar{\rho} = 0 \) in the initial queries. At some point, a stage is reached for which a greedy policy becomes more desirable, because it reaches the termination criterion \( \rho(t) < \bar{\rho} \) earlier than a \( \bar{\rho} = 0 \) optimal policy characterized by the conditions in Corollaries 7 and 8.

A final consideration for the case in which \( \bar{\rho} > 0 \) involves the initial condition. The values that conditional existence probability \( \rho(t) \) take all depend explicitly on the initial existence probability \( \rho_0 \). This sensitivity should be explored in analyses or execution of searches that employ this termination criterion.

3.7 Summary of ISIS Search Contributions

The growth of online extremism has created the need for capabilities to mitigate the threat posed by the abusive or threatening behavior of these extremist users. In this work we have developed a set of capabilities which allow for more effective mitigation of these threats. These capabilities can be used to enhance the performance of law enforcement or other entities that are responsible for protecting the public from online extremist groups. Our approach combined statistical modeling of extremist behavior with optimized search policies. Our behavioral modeling allowed us to predict new extremist users, determine if two accounts belong to the same extremist user, and predict the network connections of suspended extremist users when they create new accounts. We used our behavioral models to formulate a network search policy to find the new accounts of suspended
extremist users when they return to the social network. Simulations based on actual ISIS users found that our policy was much more efficient than other benchmark approaches.

While our analysis focused on terrorist extremist groups such as ISIS in the social network Twitter, the capabilities we developed can apply to any online extremist group and any social network. Nothing in our modeling or search policy is specialized to ISIS or Twitter. Users that engage in some form of online extremism or harassment will have very similar behavioral characteristics in social networks. They will connect to a specific set of users which form their extremist group. They will create new accounts which will resemble their old accounts after being suspended. When they return to the social network after being suspended, they will reconnect with certain former friends with higher probability. In addition, all of our capabilities do not require the cooperation of social network operators. Therefore, all the capabilities we developed here are agnostic to the extremist group and social network.

In Chapter 4, we develop a method for collecting a social media data set that consisting of the members of an identifiable group that exhibits some homophily, with an emphasis on groups of users associated with geographic locations.
Chapter 4

Building a Location-Based Set of Social Media Users

The difficult problem of identifying a social media user’s location has many applications and has been the subject of much inquiry. This chapter addresses a problem that is closely related to the social media user location identification problem. Instead trying to determine the locations of users in a given dataset, we start with a location, or other grouping, and seek to build a dataset of social media users in that location or group. Our effort makes use of many of the principle findings and ideas from previous research in the area of social media user location-identification and community detection.

The development in this chapter will focus primarily on collecting users belonging to a specific location, but we provide an example in which apply this method to collect users belonging to an ideological group at the end of the chapter.

4.1 Background and Motivation

A method that produces the set of social media users in a specific location has applications across multiple domains. Consider a businesswoman looking to open a retail business in Binghamton, New York. In order to learn about her prospective customer base and the local business climate, she might conduct Internet searches, read local media publications, and even lookup public data sources such as tax and census data. What if, in addition to these sources of information, she had a set of all of the Twitter users in the greater Binghamton area, and therefore had access to all of their publicly available tweets?
In addition to business applications, this method might be useful to political campaigns, which often conduct polls and surveys to better understand what issues are important to people in certain areas. Observing the social media content of all of the users in the targeted locations could provide another perspective and catch anomalies that traditional surveys might miss.

Another application deals with community health and tracking of dangerous infections. Consider a situation in which a dangerous disease is reported in a remote location (e.g., see Yesica Fisch, Associated Press [149]). For a multinational, nongovernmental, or governmental organization, monitoring such an outbreak could be difficult and costly. If an agency had lists of all of the Twitter users in the potentially affected areas, observing the content people are posting could be useful in ascertaining whether the disease is spreading.

Many research efforts have investigated how social media data can be used in responses to emergencies and natural disasters. Almost all of these deal with collection and analysis of real time data. However, if emergency responders were able to establish a list of social media users that live in a disaster area after a disaster has occurred, even if the users were not actively posting content during of after the disaster, they might be able to obtain much more focused information. Examining these users’ timelines before, during, and after the disaster event could serve as a powerful aid in assessing the event’s impact, and they could even attempt to make contact with users in this set directly through the social media application.

A final application of our method deals with the tracking of unrest and violence. Could the Twitter content coming from Corinto, Colombia be useful in determining whether the Revolutionary Armed Forces of Colombia (FARC) is active in the area (see Figure 4-1)? Would the tweets from users in Caracas, Venezuela enable a better understanding of the political violence there [48], and perhaps give insight into a possible solution?

All of these applications serve as our motivation. Our primary contributions in this chapter are our expand—classify approach to social media data collection and our adaptation of the Ising model for use in a social media user classification context. We detail a few implementations which provide interesting and potentially useful applications of the methodology we developed. We conduct all of our implementations in Twitter, though our method could generalize to other social media applications and sites as well. Twitter users have the option to indicate their locations in self-declared profile information fields. Of these, our implementations look for location cues in each user’s location, description, name, and screen name fields. Our method also uses Twitter relationships in performing location classification. In Twitter, these relationships are directional, where one user is
said to “follow” another. A user $i$ is said to have “followers” and “friends,” with the latter being the set of other users $i$ is following.

One of the difficulties inherent in determining social media user locations is the problem of obtaining a labeled dataset. In order to evaluate the performance of the method we present, we provide several implementations using Twitter data in which geotagged tweets serve as labeled data. We also use this labeled data to train and validate model parameters, although we show that our method can perform well even without fitting parameters to labeled data. Outside of using geotagged tweets for these tasks, our method does not make use of user content (i.e., tweets) in location classification.

Finally, it is important to consider the privacy and ethical aspects in using social media data. The goal of this research is to construct a set of social media users in a certain location so that their publicly available content can be collected, aggregated, and analyzed for interesting structures and trends. Recent efforts to identify or track of specific individuals through social media accounts have brought to light ethical and privacy implications of social media data mining and resulted in some public scrutiny (e.g., see Marotti [97], Bromwich et al. [25], or Timberg and Dwoskin [135]). We aim to avoid these ethical implications by focusing on the utility of insights gained from aggregated data gathered from a specific group of users, rather than the identification and tracking of specific individuals.
4.1.1 Previous Work

The problem of accurately identifying a social media user’s location has been a topic of interest in recent social media research. This problem has many real-world applications, including those in emergency response, marketing, law enforcement, military intelligence, and anti-terrorism.

Bo et al. [20] approach the user location problem using text classification. This method relies solely on user content to identify a user’s location, for the purpose of enhancing a user’s experience with an application. Han et al. [68] take a similar approach that also relies using location indicative words from a user’s content and profile information in a supervised classification model. These methods have been shown to correctly classify 50%–60% of user locations on test data. A downside to using these methods is that they require collection and parsing of each user’s posted content, which can become computationally expensive. They also do not use relationship data in classifying locations.

Other approaches to the user location problem use social network connections in location classification. Backstrom et al. [9] provides a detailed analysis of how distance correlates to online relationships in social media sites. One of the important findings in this work is that relationships tend to be less geographically localized in more dense population centers. We make the same observation, and our ability to build sets of users from large metropolises suffers as a result of this characteristic.

Davis Jr. et al. [41] give a method for inferring Twitter user location that uses declared profile locations, tweet geo-locations, and the locations of each user’s friends. This approach first attempts to classify users to a fixed set of cities based on the users’ content and profile information. Using these inputs the authors could classify a subset of users. Users that could not be classified by their content and profile information were then classified to the most frequently occurring relationship among their friends. Davis Jr. et al. [41] limit “friendships” to reciprocal relationships and do not use relationship data to classify users with too many or too few friends.

Compton et al. [35] take a similar approach to Davis [40], using Twitter user mentions of other users to form a social network, and using connections in this mention network to make classifications based on the known locations of connected users. The authors use a global optimization to simultaneously assign grid location to all users from whom location is unknown, using a coordinate descent approach to perform variational minimization, and find that they can geo-locate over 80% of tweets. In order to evaluate their method the authors restrict their data to users from whom
geotagged content is available and does not conflict with their declared location.

Many others have adopted similar approaches to identifying user locations from a social media data set. Jurgens [77] uses a label propagation algorithm to assign locations to users in a social network based on a few known user locations. Kong et al. [85], propose methods that assign weights to user relationships that quantify their utility in discerning location. McGee et al. [100] also introduce a model that uses weighted social media relationships to determine user locations, based on a model for predicting online relationship strength proposed by Gilbert and Karahalios [58]. Li et al. [92] give a method for assigning locations based on user behavior likelihood models, and Rout et al. [123] show that support vector machines can also be used to classify users’ locations.

There have also been many efforts in community detection within social media. Community detection can be thought of as a generalization of the user location problem, and approaches to the two problems often rely on similar assumptions. Like location classification methods, community detection methods typically employ similarities in user profile features and content, as well as user relationships, when assigning community membership.

Leskovec and Mcauley [90] present a well-known community detection algorithm, which the authors refer to as finding social circles. This algorithm considers both group interconnectedness and similarity in features. Another well-known approach to community-detection are mixed-membership stochastic blockmodels, introduced by Airoldi et al. [3], which rely solely on network structure to assign users into community groups. Both of these community detection methods are unsupervised. This is in contrast to most location classification methods, which tend to rely on having a set of labeled data.

While many of the user location classification methods in the literature cite online advertising and customized user experience as their motivation for learning user location, some efforts in this area of research have their roots in emergency response, crises, and situational awareness. Starbird et al. [132] uses collaborative filtering and support vector machines to identify Twitter users that are physically present at mass disruption events, such as the Occupy Wall Street protests in New York City in 2011. Kumar et al. [86], working in a similar vein, introduce a method of identifying users that are providing useful information for gaining situational awareness on the Arab Spring movements in the same year. This approach combines topic models with user location information to determine user relevance.
4.1.2 Expand—Classify Methodology

Unlike many previous efforts, our goal is to build a comprehensive set of users in a specific location, with an emphasis on small or medium-sized population centers. Our approach is to apply an expand—classify methodology, illustrated in Figure 4-2. Each iteration begins with a set of social media users. Each user in the dataset has been classified as either being in the location of interest, or not in the location of interest. In the expand step, a subset of the users that have been classified as being in the location of interest is selected for expansion queries. Some or all of the social network connections of these users are collected and added to the data set. This collection not only discovers new users to add to the dataset, but also identifies previously unknown connections between users that are already in the user data set.

In the classify step, the location classification of each user is updated. Similar to Compton et al. [35], we perform a global optimization to classify each user’s based on both the user’s account information and the classifications of the user’s connections. In order to carry out this classification, we construct a simple but powerful factor graph model of the social network and find the maximum likelihood according to this model.

Our approach relies to two very general assumptions.

1. We assume that social media users in the target location (or, more generally, target group) tend to connect with each other at a higher rate than users in different locations. We refer to this phenomenon as location homophily. This assumption is supported by the findings of multiple research efforts, including Backstrom et al. [9].
2. We assume that certain user profile characteristics, such as the use of location-specific words or phrases, can serve as useful features in classifying whether the user is in the target location (or group). This assumption is supported by the work of Han et al. [68] and others.

The first assumption is embedded in our collection methodology. By collecting the friends and followers of users that are identified as being in the target location during the expand step, new users in the target location will continue to be identified. Our classification model makes use of both of these assumptions.

4.2 Classification Model

Each iteration of the process depicted in Figure 4-2 requires updated location classifications of all users collected in the data set. In this section, we propose a classification method that uses a factor-graph model of the social network. This factor graph model is based on the image segmentation model presented by Zabih and Kolmogorov [151] and is closely related to the Ising energy model.

4.2.1 Factor Graph Representation

We use a factor graph to serve as a generative model of user locations and connections within social media. Nodes in a factor graph represent variables, which can be latent or observed. Nodes are connected to factors, which imply a dependency structure that specifies a factorization of the joint distribution function of variables associated with the nodes.

Nodes.

In our graph we consider three types of nodes, representing the three types of variable in our model:

1. User profile information $x_i$ for each user $i$. This vector includes information on whether a user’s profile information contains location-specific terms. These values are observed.

2. Relationship features $z_{i,j}$ for pairs of users $i,j$. This vector encodes the nature of the social media relationship (who is following whom), the out degree of the “follower,” and the in-degree of the “friend.” These values are also observed.

3. User location class $\ell_i$ for each user $i$. These values are unobserved, or latent.
Factors.

The factor graph also contains two types of factors:

1. For each user $i$, corresponding nodes $x_i$ and $\ell_i$ share a common factor with potential

   \[ f(x_i, \ell_i) = e^{-\phi(x_i, \ell_i)}. \]

2. For each pair of users $i, j$, the corresponding nodes $\ell_i, \ell_j$, and $z_{i,j}$ share a factor with potential

   \[ g(z_{i,j}, \ell_i, \ell_j) = e^{-\psi(z_{i,j}, \ell_i, \ell_j)}. \]

For pairs of users $i, j$ for which there is no observed social media relationship (encoded in vector $z_{i,j}$), we fix $g(z_{i,j}, \ell_i, \ell_j) = 1$. This modeling choice and the assumptions implied by it are discussed in Section 4.2.2 below.

![Figure 4-3: Factor graph model for social media user location classes.](image)

Figure 4-3 provides a notional social network consisting of four users and the corresponding factor graph model. This figure omits trivial factors with fixed potential functions, i.e., factors
corresponding to pairs of users that are not connected to each other in the social network. The nodes with heavier outlines represent observed values, while the $\ell_i$ (location) nodes represent latent variables.

Given $N$ observed users in a social network, let $X$ be a vector of all of the users’ observed location variables, $Z$ be a vector of all of the observed link strengths in the social network, and $L$ be a vector of user location classes. Our factor graph model implies that the joint probability of these vectors is proportional to the product of the potentials, i.e.,

$$P(X, Z, L) = \prod_{i=1}^{N} e^{-\phi(x_i, \ell_i)} \prod_{\{i, j: i < j\}} e^{-\psi(z_{i,j}, \ell_i, \ell_j)} \frac{Z(\Phi, \Psi)}{Z(\Phi, \Psi)},$$

(4.1)

where $Z(\Phi, \Psi)$ is the partition function. We refer to $\phi$ and $\psi$ respectively as the profile energy and link energy functions.

4.2.2 Model Characteristics

We have assumed that user profile characteristics and relationships can be modeled by a probability distribution that factorizes according to the structure of our factor graph representation. We now provide some additional specifications that support our objective of obtaining a set of users in a specified location and follow from our assumptions.

Location Classes.

We assume a two-class location model, in which we only wish to label each user in the dataset as either being in the location of interest or not in the location of interest. We define location class

$$\ell_i = \begin{cases} 
1 & \text{User } i \text{ is in location of interest} \\
0 & \text{User } i \text{ is not in location of interest.}
\end{cases}$$

Link Energy Function.

Without yet specifying a link energy function, we impose the following limitations on its structure:

1. We assume that the probability of a social network relationship forming between two users in social media is small, irrespective of whether or not they are in the same location. To demonstrate the implications of this assumption, let $A_{ij}$ be the event that a relationship
exists between user $i$ and user $j$ on a social media site. While the likelihood ratio
\[
\frac{\mathbb{P}(A_{ij} | \ell_i \neq \ell_j)}{\mathbb{P}(A_{ij} | \ell_i = \ell_j)}
\]
could be very large or very small, our assumption implies that
\[
\frac{\mathbb{P}(A_{ij} | \ell_i \neq \ell_j)}{\mathbb{P}(A_{ij} | \ell_i = \ell_j)} \approx 1.
\]
Based on this assumption, we set $\psi(z_{ij}, \ell_i, \ell_j) = 0$ when there is no observed relationship between user $i$ and user $j$. Our fixing of the link potential $g(z_{ij}, \ell_i, \ell_j = 1$ in the absence of any observed relationships between user $i$ and user $j$ follows directly from this assumption.

2. For any users $i$ and $j$, the following inequalities hold:
\[
\psi(z_{ij}, 1, 1) \leq \psi(z_{ij}, 0, 0) \leq \psi(z_{ij}, 0, 1) \\
\psi(z_{ij}, 1, 1) \leq \psi(z_{ij}, 0, 0) \leq \psi(z_{ij}, 1, 0).
\]
Essentially, we assume that social network links between two users that are in different location classes are always less probable than links between users in the same location class. We allow for a positive cost to be associated with classifying two connected users as both being outside of the location of interest, as this classification does not necessarily imply that they are in the same location. However, we restrict this cost to be no more than the cost of assigning different location classes to a pair of connected users. These inequalities follow from our assumption on location homophily. By convention, we set
\[
\psi(z_{ij}, 1, 1) = 0.
\]
An important implication that follows from this assumption is
\[
\psi(z, 1, 1) + \psi(z, 0, 0) \leq \psi(z, 1, 0) + \psi(z, 0, 1).
\]
These inequalities allow for efficient energy minimization using graph cuts, as shown by Kolmogorov and Zabin [84].
4.2.3 Classification Optimization

We set as our objective to find the most probable location classifications, \( L \). More formally, having observed values \( X \) and \( Z \), we seek a solution to the following optimization problem:

\[
\text{maximize } \left[ \mathbb{P}(X, Z, L) \right],
\]

which is equivalent to finding a solution to the following:

\[
\text{minimize } L \left[ \sum_i \phi(x_i, \ell_i) + \sum_{i<j} \psi(z_{i,j}, \ell_i, \ell_j) \right]. \tag{4.2}
\]

Boykov et al. [22] provide an efficient method for minimizing energy functions of this nature using graph cuts. Following their method, and consistent with the subsequent findings by Kolmogorov and Zabin [84], we construct a directed graph consisting of the following nodes:

- A source node \( s \).
- A sink node \( t \).
- A node \( u_i \) for each user \( i \).

We add the following edges and capacities:

- An edge from each user node \( u_i \) to the sink node \( t \) with capacity

\[
c_{(u_i, t)} = \phi(x_i, 1).
\]

- For each pair of users \( i, j \) for which there is an observed relationship in the social network, edges from \( u_i \) to \( u_j \) and from \( u_j \) to \( u_i \) with capacities

\[
c_{(u_i, u_j)} = \psi(z_{i,j}, 1, 0) - \frac{1}{2} \psi(z_{i,j}, 0, 0),
\]

\[
c_{(u_j, u_i)} = \psi(z_{i,j}, 0, 1) - \frac{1}{2} \psi(z_{i,j}, 0, 0).
\]

- An edge from the source node \( s \) to each user node \( u_i \) with capacity

\[
c_{(s, u_i)} = \phi(x_i, 0) + \frac{1}{2} \sum_{\{j: j \neq i\}} \psi(z_{i,j}, 0, 0).
\]
We refer to this graph as the **Energy Graph** representation of the energy function

\[
E(L) = \sum_i \phi(x_i, \ell_i) + \sum_{i<j} \psi(z_{ij}, \ell_i, \ell_j),
\]

which is the objective function in the classification optimization, equation (4.2). Now we show that performing the optimization in equation (4.2) is equivalent to finding the minimum capacity cut on the Energy Graph, which can be done efficiently using minimum cut—maximum flow algorithms [22].

Figure 4-4: Energy graph representation of the energy equation corresponding to the factor graph in Figure 4-3.

**Theorem 9** (Minimum Cut—Classification Optimality Equivalence). *Given an energy minimization of the form given in (4.2), the optimal solution corresponds exactly to a minimum capacity s-t cut in the corresponding Energy Graph representation.*

Before proving Theorem 9, it is useful to establish our notation and provide an important Lemma. We let \( G = (\mathcal{V}, \mathcal{E}) \) be the Energy Graph representation of for an energy function of the form in equation (4.3), which adheres to the assumptions in Section 4.2.2, where \( \mathcal{V} \) is the set of nodes and \( \mathcal{E} \) is the set of edges. A valid s-t cut is a partition of \( \mathcal{V} \) into two subsets: \( S \) and \( T \), where source node \( s \in S \) and sink node \( t \in T \). We define the cut-set \( C \subset \mathcal{E} \) as the set of directed
edges going from any node in set $S$ to any node in set $T$. For arbitrary location classifications $\mathbf{L} = (\ell_1, \ell_2, \ldots, \ell_N) \in \{0,1\}^N$, consider the $s$-$t$ cut on the Energy Graph $\mathcal{G}$ that partitions the nodes according to their location classes. We denote the set of nodes in the subset belonging to the source node as $S_L$, where node $u_i \in S_L$ for all users $i$ for which $\ell_i = 1$. Likewise, $u_j \in T_L$ for all $j$ such that $\ell_j = 0$. We refer to this cut as the $L$-configuration cut on the Energy graph.

We state our Lemma.

**Lemma 2 (Graph Equivalence).** Suppose we are given an energy function of the form

$$E(\mathbf{L}) = \sum_i \phi(x_i, \ell_i) + \sum_{i<j} \psi(z_{i,j}, \ell_i, \ell_j),$$

where functions $\phi(x_i, \ell_i)$ and $\psi(z_{i,j}, \ell_i, \ell_j)$ adhere to the assumptions in Section 4.2.2. Then, for arbitrary location classification vector $\mathbf{L} \in \{0,1\}^N$, the value of the function $E(\mathbf{L})$ is equal to the capacity of the $L$-configuration cut on the corresponding energy graph.

**Proof.** Proof of Lemma 2. Let $G = (\mathcal{V}, \mathcal{E})$ be the energy graph representation of energy function $E$, and consider an arbitrary fixed location classification vector $\mathbf{L} \in \{0,1\}^N$. From our definition of the $L$-configuration cut, $\ell_i = 1$ implies node $u_i \in S_L$ and $(u_i, t) \in C_L$. Likewise, $\ell_i = 0$ implies $(S, u_i) \in C_L$. Now consider an arbitrary pair of user nodes $u_i, u_j$, $i \neq j$. One of the following cases apply: (Case 1) both users are assigned to location class 1 ($\ell_i = \ell_j = 1$), (Case 2) one of the users is in location class 1 and the other is assigned to location class 0 ($\ell_i \neq \ell_j$), or (Case 3) both users are assigned to location class 0 ($\ell_i = \ell_j = 0$). We consider the implications of each case on cut-set $C_L$.

**Case 1.** Because $u_i$ and $u_j$ are both in set $S_L$, edges between these two nodes are not in $C_L$. It follows that

$$\ell_i = \ell_j = 1 \Rightarrow (u_i, u_j), (u_j, u_i) \notin C_L.$$ 

However, edges $(u_i, t)$ and $(u_j, t)$ are in the cut-set $C_L$. Figure 4-5 provides an illustration of this case.

**Case 2.** Without loss of generality, assume $\ell_i = 1$ and $\ell_j = 0$, implying $(u_i, t) \in C_L$ and $(s, u_j) \in C_L$. Because $u_i \in S$ and $u_j \in T$, edge $(u_i, u_j)$ is also in the cut-set $C_L$. This case is depicted in Figure 4-6.

Note that the reverse edge from $u_j$ to $u_i$ in Figure 4-6 is *not* in the cut-set because they go from set $T_L$ to set $S_L$. 

141
Figure 4-5: Illustration of Case 1 cut. Nodes in set $S_L$ are shaded green, while nodes in set $T_L$ are shaded red. Dashed edges are in cut-set $C_L$.

Figure 4-6: Illustrations of both minimum cut possibilities for Case 2. Nodes in set $S_L$ are shaded green, while nodes in set $T_L$ are shaded red. Dashed edges are in cut-set $C_L$.

**Case 3.** Finally, we consider the case in which $\ell_i = \ell_j = 0$. This case is very similar to Case 1: edges $(s, u_i)$ and $(s, u_j)$ are in $C_L$, while other edges incident to these nodes are not in the cut-set (see Figure 4-7).

From these rules we can identify all of the edges comprising set cut-set $C_L$ for an arbitrary
location class vector $L$. The total capacity of the cut is the sum of all of these edge capacities:

$$
\sum_{a \in C_L} c_a = \sum_{i : \ell_i = 0} c_{(s, u_i)} + \sum_{i : \ell_i = 1} c_{(u_i, t)} + \sum_{i < j : \ell_i = 1, \ell_j = 0} c_{(u_j, u_i)} + \sum_{i < j : \ell_i = 0, \ell_j = 1} c_{(u_i, u_j)}
$$

$$
= \sum_{i : \ell_i = 0} \left( \phi(x_i, 0) + \frac{1}{2} \sum_{j : \neq i} \psi(z_{i,j}, 0, 0) \right) + \sum_{i : \ell_i = 1} \phi(x_i, 1)
+ \sum_{i < j : \ell_i = 1, \ell_j = 0} \left( \psi(z_{i,j}, 1, 0) - \frac{1}{2} \psi(z_{i,j}, 0, 0) \right) + \sum_{i < j : \ell_i = 0, \ell_j = 1} \left( \psi(z_{i,j}, 0, 1) - \frac{1}{2} \psi(z_{i,j}, 0, 0) \right)
$$

$$
= \sum_{i=1}^N \phi(x_i, \ell_i) + \sum_{i < j : \ell_i = \ell_j = 0} \psi(z_{i,j}, 0, 0) + \sum_{i < j : \ell_i = 1, \ell_j = 0} \psi(z_{i,j}, 1, 0) + \sum_{i < j : \ell_i = 0, \ell_j = 1} \psi(z_{i,j}, 0, 1)
$$

$$
= \sum_{i=1}^N \phi(x_i, \ell_i) + \sum_{i < j} \psi(z_{i,j}, \ell_i, \ell_j) = E(L).
$$

Proof. Proof of Theorem 9. Theorem 9 follows almost immediately from Lemma 2. Suppose we find the minimum $s$-$t$ cut on the Energy Graph corresponding to energy function $E(L)$, and let sets $S$ and $T$ be the corresponding partition of $\mathcal{V}$ and $C \subset \mathcal{E}$ be the cut-set. In order for the cut to be valid, each user node $u_i$ must be in either set $S$ or set $T$. For each user node $u_i \in S$, edge $(u_i, t)$ is in the cut-set $C$ and we set $\ell_i = 1$. For each user node $u_i \in T$, edge $(s, u_i)$ is in $C$ and we set $\ell_i = 0$. Let $L^*$ be the resulting vector of location assignments.

From Lemma 2, the capacity of this cut is $E(L^*)$. Because the sets $S$ and $T$ were constructed
from the minimum capacity $s$-$t$ cut on the graph, there cannot be another location assignment $L$ for which $E(L) < E(L^*)$, as such a vector would allow for the construction of an $s$-$t$ cut with lower capacity.

As a final note on the classification model, we consider the case in which the profile energy function exhibits the following symmetry:

$$\psi(z_{i,j}, 0, 1) = \psi(z_{i,j}, 1, 0).$$

In words, the cost of classifying user $i$ to location 0 and user $j$ to location 1 is the same as the cost of classifying user $i$ to location 1 and user $j$ to location 0. Our implementations in the following sections exhibit this property. In this case, the directed Energy Graph can be replaced by an undirected graph without affecting the above result. Minimization of the energy function in equation (4.2) is equivalent to finding the minimum cut on the undirected Energy Graph representation [151].

4.3 Choosing the Energy Functions

We now discuss our choices for the energy functions we use in our implementations of this methodology. The input functions $\psi$ and $\phi$ are important: they quantify the trade-off between the value of the information contained in a user’s profile (e.g., the user’s self-identified “location”) and value of the user’s social connections. For our implementations, we use a fixed link energy function that follows from the findings of Backstrom et al. [9], McGee et al. [100], and Davis [40]. For the profile energy function, we compare two approaches: a naive approach in which we fix the profile energy based on qualitative observations, and a parametric approach in which we fit a probabilistic model to a subset of the data for which location labels are available.

4.3.1 Link Energy Model

Social media research has consistently shown that users tend to connect to other users with whom they have an existing relationship outside of social media (see, e.g., Backstrom et al. [9], Davis Jr. et al. [41]) and that relationships between users with lower degrees tend to be indicative of closer relationships [58]. Based on these findings we assume that the utility of a social media relationship in inferring that two users belong to the same location decreases as the number of relationships (or degree) of linked users grows. If we observe that a user is following a superstar with millions of social
media connections, for example, we would not consider that online relationship to be very valuable in location inference. On the other hand, if two users are connected and each has only a total of 20 online connections, we consider that relationship to be indicative of an existing relationship outside of social media, which could mean that the users live in close proximity to each other.

Suppose user 1 is following user 2 in Twitter. We also observe that user 1 follows a total of \( z_1 \) other users and that user 2 has a total of \( z_2 \) followers. We encode this information in vector \( z_{1,2} \), and use a sigmoid function to model our intuition on link energy, setting

\[
\psi(z_{1,2}, 1, 0) = \psi(z_{1,2}, 0, 1) = \frac{\gamma}{1 + \exp(-2 + (2/\alpha_1)z_1 + (2/\alpha_2)z_2)}.
\]

Using this form, the parameters \( \gamma, \alpha_1, \) and \( \alpha_2 \) all have useful interpretations. Parameter \( \gamma \) is the link energy of the closest relationships. If user 1 has very few friends and user 2 as very few followers, this function approaches \( \gamma + \exp(-2) \approx \gamma \). The parameters \( \alpha_1 \) and \( \alpha_2 \) are the numbers of friends and followers, respectively, that would result in half this link energy.

Based on the findings of McGee et al. [100] and our own investigation of Twitter relationships, we fixed \( \alpha_1 = 500 \) and \( \alpha_2 = 5000 \). We have observed that users with more than about 500 friends tend to be connected to more celebrities, politicians, and media sites, while users with more than about 5000 followers tend to start having more than just a local following. Figure 4-8 illustrates how this function decays as the degree of each node in a social media relationship increases. This figure illustrates the energy of a directed relationship in which user 1 is following user 2. In the left-hand plot, user 1’s out-degree, or friends count, is fixed at 20. We see that if user 2 has close to 0 followers, the link energy is close to \( \gamma \), but decays as the number followers grows. The right-hand plot shows the same effect as the number of user 1’s friends increases, while user 2’s follower count is fixed at 20.

The function \( \psi(z_{ij}, 1, 0) \) can be interpreted as a log likelihood ratio. Recall that we have set \( \psi(z_{ij}, 1, 1) = 0 \).

\[
\psi(z_{ij}, 1, 0) = \log e^{\psi(z_{ij}, 1, 0)}
= \log \left( \frac{1}{e^{-\psi(z_{ij}, 1, 0)}} \right)
= \log \left( \frac{e^{-\psi(z_{ij}, 1, 1)}}{e^{-\psi(z_{ij}, 1, 0)}} \right).
\]

This is the log ratio of factor potentials from our factor graph model. This ratio can be thought of as
The log likelihood that an observed relationship is indicative of two users sharing the same location. This interpretation is useful in considering our choice for the parameter \( \gamma \). In our implementations, we initially set \( \gamma = \log(5) \), which implies that low-degree relationships are about five times more likely to share a common location than not. We show through sensitivity analysis that this achieves good performance in many cases, although it is not often optimal.

We have addressed the link energy value for connected users when both users are in the location of interest, \( \psi(z_{ij}, 1, 1) = 0 \), and when one user is in the location of interest and the other is not,

\[
\psi(z_{ij}, 1, 0) = \psi(z_{ij}, 0, 1) = \frac{\gamma}{1 + \exp\left(-2 + \frac{(2/\alpha_1)z_1 + (2/\alpha_2)z_2}{\gamma}\right)}.
\]

We still have to address the link energy value when the pair of connected users is not in the location of interest. Because our collect—classify methodology continues to collect friends and followers from users classified within the location, we do not expect to obtain many, or perhaps even any relationships between user pairs in which both users are outside of the location of interest. By assumption,

\[
\psi(z_{i,j}, 1, 1) \leq \psi(z_{i,j}, 0, 0) \leq \psi(z_{i,j}, 0, 1) = \psi(z_{i,j}, 0, 1).
\]

Given these bounds, where we set link energy \( \psi(z_{i,j}, 0, 0) \) in this range provides for interesting discussion. On one hand, we can set \( \psi(z_{i,j}, 0, 0) = \psi(z_{i,j}, 1, 1) = 0 \), arguing that users assigned to the same location class should always have zero link energy. However, this fails to recognize that unlike user pairs in location class 1, two users in location class 0 do not necessarily share the

---

**Figure 4-8:** Decay of link energy \( \psi(z_{1,2}, 1, 0) \) as the number of user 1 friends or number of user 2 followers increases.
same geographic location. It follows that relationships between two users in location class 0 should be associated with some positive energy, implying that they are less probable than relationships between users in location class 1.

Therefore, we assume that the link energy between users in location class 0 is very close to the link energy between users in different location classes. Specifically, we set

$$\psi(z_{i,j}, 0, 0) = \lambda \psi(z_{i,j}, 1, 0),$$

where $\lambda$ is close to, but less than 1. We provide sensitivity analysis of this decision using other values of $\lambda \in [0, 1]$.

We can think of $\lambda$ as a way of dampening the effect of relationships with users in location class 0. If $\lambda = 0$, then a user $i$ with many strong relationships with other users in location class 0 will be “pulled” into location class 0 with them. If $\lambda = 1$, user $i$’s connections with users in location class 0 will not have a direct affect on $i$’s classification, because these relationships will result in the same link energy in either case.

### 4.3.2 Profile Energy Function

We adopt two different approaches for coming up with a profile energy function. In the first case, we assume that there is no labeled data available. In this case, an analyst can rely on observations, expert information, or intuition to construct a simple and yet potentially powerful profile energy model. Alternatively, if there is some labeled data available or if a feasible method exists for labeling some of the data as it is collected, an analyst can fit a parametric probability model to the labeled data and use this model to construct a link energy function.

Just as the link energy function has a probabilistic interpretation as a log likelihood ratio, the link energy function has an analogous interpretation. Specifically,

$$\phi(x_i, 0) - \phi(x_i, 1) = \log \frac{e^{-\phi(x_i, 1)}}{e^{-\phi(x_i, 0)}}.$$

Because the factor potentials can be scaled without affecting the joint probability distribution—the scaling will be absorbed into the partition function in equation (4.1)—only the difference between location potentials, and not the values themselves, are relevant in optimizing user $i$’s location assignment. Rescaling the energy functions is equivalent to adding a constant to the objective function in equation (4.2). Therefore, when determining a profile energy function, one need only
be concerned with the difference, \( \phi(x_i, 0) - \phi(x_i, 1) \) for each user \( i \), recognizing that this difference represents the log likelihood ratio of the location classes.

**Fixed Profile Energy Model**

In the absence of labeled data, an analyst could use intuition, expert information, or observations to produce a simple odds table, from which the profile energy function could be produced. For example, suppose the analyst is interested in finding Twitter users in Boston, Massachusetts. The analyst might decide observing the word “Boston” in a Twitter user’s profile location field is a useful feature in this classification. Therefore, the analyst could simply conjecture odds for each feature category, such as those given in Table 4.1.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Odds (location:non-location)</th>
<th>( \phi(x_i, 0) - \phi(x_i, 1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile loc. includes “Boston”</td>
<td>20:1</td>
<td>( \log(20) )</td>
</tr>
<tr>
<td>Profile loc. does not include “Boston”</td>
<td>1:10</td>
<td>( -\log(10) )</td>
</tr>
</tbody>
</table>

In practice we found that this relatively naive approach to constructing a link energy function achieves performance similar to the parametric approach described below.

**Parametric Approach**

The drawback of the naive approach is that it requires an analyst to come up with a set of location features and an location odds table based on those features. This task can become very difficult as the number of features increases. However, there might be very many features that are useful in user location classification, and these features might not be mutually exclusive. In this case, a parametric model would be useful in constructing the profile energy function. Because of its simplicity, we propose a linear model:

\[
\phi(x_i, 0) - \phi(x_i, 1) = \beta^T x_i.
\]

Using our interpretation of the profile energy difference, \( \phi(x_i, 0) - \phi(x_i, 1) \), as a log likelihood ratio, this linear model is the well-known logistic regression model and is easily fit on a set of labeled data using existing methods and open source software packages. We fit a regularized logistic
regression model, which finds parameters $\beta$ by performing the following optimization:

$$
\maximize_{\beta} \left[ C \left( \sum_{i: \ell_i=1} \log \left( \frac{1}{1 + \exp(-\beta^T x_i)} \right) + \sum_{j: \ell_j=0} \left( \frac{1}{1 + \exp(\beta^T x_j)} \right) \right) - \|\beta\| \right],
$$

where $C$ is the regularization parameter and $\|\beta\|$ is the regularization norm ($\sum |\beta|$ for $L_1$ regularization and $\frac{1}{2} \beta^T \beta$ for $L_2$ regularization) [111].

### 4.4 Implementations

In this section we provide an analysis of the results obtained by applying this method to collect a set of users from several locations:

- Corinto, Colombia.
- Casimiro de Abreu, Brazil.
- Caracas, Venezuela.

For each location we provide a brief overview, a summary of the composition of the seed set, implementation details, and an analysis of the results obtained. We use the area under the Receiver Operator Characteristic curve (AUC) as our metric of performance. We close this section by discussing the challenges we found when applying this method to larger population centers, and propose methods for overcoming them.

In order to obtain a set of seed users for each location, we used the Twitter “user search” API which allows queries for users meeting certain criteria [140]. In some cases, this method did not return any results for a specific location, and more a more general location query string was used. The nature of the resulting seed set will be provided for each location.

#### 4.4.1 Corinto, Colombia

Corinto, Colombia is a town in the Cauca district of Colombia, located about 30 miles southeast of Cali. Including the population of its nearby and larger neighbor Miranda, the Corinto area has a population of approximately 30,000 people. Using Google Maps [64], we located its center at 3.174159, 76.25880. For labeling geotagged tweets, we used a radius of 7 miles (see Figure 4-9). This radius includes Miranda as well as some of the smaller nearby towns but does not include any part of the Cali metropolis.
Seed Set

Querying the Twitter user search API for “Corinto, Colombia” to obtain a seed user set for this implementation did not return any results. Instead, we obtained 128 users returned from running the individual queries “Corinto” and “Colombia” in the user search API. Of these results, 67 profiles contained the word “Corinto” in the location, description, name, or screen name, while 62 of the results contained the word “Colombia” in at least one of these four fields. Only one result contained both strings.

Searching through the tweets from these accounts yielded 12 geo-located tweets; of these only one was inside the 7 mile radius depicted in Figure 4-9. These locations were consistent with a manual inspection of the seed accounts, which included profile locations from varying locations throughout Colombia and from around the world. The seed set appeared to contain very few accounts in the target location.
Logistic Regression Energy Model

In order to fit a logistic regression model on the data, we developed a method of extracting features from user profiles that might be useful in predicting the user’s location classification. First, we created two lists of character strings, $W_1$ and $W_2$, which we compared to each user’s profile information. List $W_1$ was comprised of strings that we thought might indicated a user was associated with the target location, while $W_2$ contained strings that would suggest a user was not associated with the target location. Each string from list $W_1$ was used to generate four binary feature variables, corresponding to the user profile’s location, description, name, and screen name fields. The character strings that comprise list $W_1$ for Corinto are in Table 4.2.

Table 4.2: List of character strings $W_1$ used to extract profile features for Corinto logistic regression.

<table>
<thead>
<tr>
<th>“Corinto”</th>
<th>“Cauca”</th>
<th>“Colombia”</th>
<th>“Miranda”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Corinto Colombia”</td>
<td>“Corinto, Colombia”</td>
<td>“Miranda, Colombia”</td>
<td>“Miranda Colombia”</td>
</tr>
<tr>
<td>“Corinto Cauca”</td>
<td>“Corinto, Cauca”</td>
<td>“Miranda, Cauca”</td>
<td>“Miranda Cauca”</td>
</tr>
</tbody>
</table>

If the string “Corinto” appeared in a user profile description, for example, then the feature variable corresponding to that character string in the description field would take value 1. Otherwise, the feature variable would be set to 0. Because there are 12 character strings in this list, there were 48 corresponding binary variables in the logistic regression model. We include strings containing the location Miranda, Colombia in list $W_1$ because Miranda is a population center within the 7-mile radius of Corinto.

List $W_2$ simply contained a list of world cities with populations over 1,000,000 from MaxMind.com [99]. This list is contained in Appendix A. We used list $W_2$ to generate four additional binary variables: one for each user profile field (location, description, name, and screen name). If any of the character strings from this list appeared in a user’s location, for example, then the binary feature variable corresponding to $W_2$ in the location field would take value 1. Otherwise, this feature variable would be set to 0.

We included five additional binary feature variables: empty location, language, UTC offset, protected account, and verified account. The empty location variable took value 1 if the user profile’s location field was left empty. If a user profile’s language was set to the local language, Spanish, then the language variable was set to 1. If the profile’s time zone matched the local UTC offset, -18000 seconds, the UTC offset variable was set to 1. The protected and verified account variables were set to match each user’s account settings, taking value 1 if the profile was protected.
or verified, respectively, and 0 otherwise.

The total number of features in the logistic regression model for Corinto was 57.

For the response variable, we used geo-located tweets posted by the users. We searched through users’ most recent posts and identified any tweets that contained geo-location data. Of these, we extracted the post that had coordinates closest to the center of Corinto (3.174159, 76.25880). If these coordinates were within seven miles of the grid coordinates at the center of the target location, the user was labeled as being inside the target location ($\ell_i = 1$). If the coordinates were outside of this radius, the user was labeled as outside of the target location ($\ell_i = 0$). Users with no geotagged tweets were not included in the logistic regression model.

In fitting the logistic regression models, we set aside some geo-located content for validation and testing. We used $L_1$ regularization and, through validation, found the model achieved the best performance using a regularization coefficient of approximately $C = 1$.

**Performance**

Beginning with the seed user set, we iterated the expand—classify approach for four hours. In the *expand* step of each iteration, we randomly selected up to 30 users from the set of users classified in the target location and queried up to 5000 of each user’s Twitter followers. We randomly selected another 30 users from the set of those classified in the target location and queried up to 5000 of each user’s Twitter friends. We used these values to expand the set of profiles efficiently while remaining within the API rate limits established by Twitter [138].

Following the collections in each iteration, all of the users in the dataset would be classified. First, the $L_1$-regularized logistic regression model was fit and validated on a randomly selected subset of the geo-located users. Using the resulting linear model as a profile energy function, all of the users in the data set were then classified by finding the minimum cut on the Energy Graph as described in Section 4.2.3. After completing this classification and, if necessary, taking a short pause to keep from exceeding the Twitter API rate limits, the friend and follower queries for the *expand* step in the next iteration would begin.

After four hours, the number of users profiles collected was 140,571. Of these, 988 were classified as being in the Corinto, Colombia target area. From the classification results, we constructed a local
probability for each user being in the target location:

\[ P_1(i) = \left(1 + \exp\left(\phi(x_i, 1) - \phi(x_i, 0) + \sum_{j \neq i} \left[\psi(z_{ij}, 1, \ell_j) - \psi(z_{ij}, 0, \ell_j)\right]\right)\right)^{-1} \]

The probability follows from our factor graph model, holding all location classifications fixed and examining the probability associated with each location class for user \( i \). Using these local probabilities, we construct a Receiver-Operator Characteristic (ROC) curve to evaluate the results on the set of points for which geo-located tweets were available. The ROC for this collection of Corinto users is plotted in Figure 4-10. The figure shows that this implementation can correctly classify about 60% of the users in the target location radius while maintaining a low false positive rate. Our method is superior to the current capability of Twitter’s user search API, which in this case was not very useful in producing even a seed set of users in Corinto, Colombia.

Figure 4-10: Corinto user classification ROC using logistic regression model for profile energy.

We attempted another method of generating users in Corinto: using the Twitter search API to find tweets from that area. One drawback of this method is that the search must include a query
string. We executed as single Twitter search for tweets originating within 7 miles of the center of Corinto, Colombia, querying for tweets containing the single character e. The query returned tweets from six unique users. However, none of the tweets returned contained geo-location data, nor were we able to find geo-located tweets among the most recent tweets in these user’s timelines. Reviewing the user profile locations for the six users revealed that two users were from Puerto Tejada, three were from Jamundí, and one was from Miranda. Of these locations, only Miranda falls inside the 7-mile radius around Corinto.

Interestingly, only one of these users, from Jamundí, appeared in the set of 140,571 we collected. This user was classified as being outside of Corinto.

**Sensitivity Analysis**

We now briefly discuss and illustrate this implementation’s sensitivity to the inputs $\gamma$, $\lambda$, as well as the logistic regression regularization. The parameter $\gamma$ serves as the magnitude of the sigmoid curve that governs the decay of link energy as the number of friends and followers increases. Higher values of $\gamma$ result in larger link energies, which cause social media relationships to have more influence over each user’s classification.

Figure 4-11 depicts the classification ROC curve plotted using several different values for $\gamma$. Of particular note is the case where $\gamma = 0$, which recovers the logistic regression classification. Comparison with this curve provides a quantification of the utility of the Twitter relationships in this classification model. Based on the AUC metric, we find that optimal performance appears to occur for higher values of $\gamma$, with $\gamma = \log(10)$ producing an AUC of 0.94. However, small variations from this value to not appear to substantially impact performance. Using only the logistic regression model ($\gamma = 0$) produces an AUC of approximately 0.64, showing that accounting for social media relationships in the model substantially improves classification performance.

The parameter $\lambda \in [0, 1]$ sets the link energy for social media relationship between users that are both classified in location set 0 (outside of the target location). We have set this parameter to 0.98, so that these relationships are approximately the same cost as relationships for which one user is in the target location and the other is not. Figure 4-12 depicts the sensitivity of the ROC curve to this value.

Higher values of $\lambda$ appear to produce the best results, and very low values performing very poorly. This poor performance results from users in the target location being misclassified as location 0 as a result of relationships with other users in location class 0. Good performance is maintained for
Figure 4-11: Sensitivity of Corinto user classification to parameter $\gamma$. Values of $\lambda > 0.75$.

Figure 4-12: Sensitivity of Corinto user classification to parameter $\lambda$. 
Finally, we investigate the model sensitivity to the logistic regression regularization. In a similar fashion to the above analyses, we fit $L1$ and $L2$ regularized logistic regression models to the geolocated training data using different values for the regularization coefficient. Without fitting the regularization coefficient through model validation, we applied the resulting linear function directly as the profile energy model. We found that the model performance was neither sensitive to the regularization norm ($L1$ vs. $L2$) nor to the regularization coefficient, except for in cases in which we significantly over-regularized the logistic regression.

Figure 4-13 shows the classification ROC using for several regularization constants for both $L1$ and $L2$ regularization. We observe that a lower value of $C$, which imply a more regularized model, results in a slight increase in performance from the value found through validation.

![ROC Sensitivity to $L1$ Regularization Coefficient](image1)

![ROC Sensitivity to $L2$ Regularization Coefficient](image2)

Figure 4-13: Sensitivity of Corinto user classification to logistic regression regularization.

**Summary of Corinto Collection**

The user collection for Corinto provides a useful example of the utility of this methodology. Using the output of a logistic regression model as a profile energy function in our factor-graph model produced a classifier with an AUC of 0.92. Tuning the parameters on test data enables an increase in performance to an AUC of 0.94.

The remainder of this section demonstrates how collections on other locations deviate from this performance. Most of our experimental location collections achieved comparable results, though in many cases logistic regression alone performed relatively well and the inclusion of the factor graph model resulted in less substantial, though often still notable, improvements.
4.4.2 Casimiro de Abreu, Brazil

Casimiro de Abreu, Brazil is a town in the Rio de Janeiro state of Brazil, located at 22.484°, 42.202°, about 80 miles east of the city of Rio de Janeiro. It has a population of approximately 35,000. For labeling geotagged tweets, we used a radius of 5 miles (see Figure 4-14). Based on the imagery available on Google Maps, there are no substantial population centers within this radius. Casimiro de Abreu falls in the coastal region of Barra de São João.

![Figure 4-14: Casimiro de Abreu, Brazil label radius, plotted on Google Maps [64].](image)

Seed Set

Querying the Twitter user search API for “Casimiro de Abreu, Brazil” did not return any results. Instead, we used 11 user profiles returned from running a user search query on “Casimiro de Abreu.” Of these results, 10 profiles contained the string “Casimiro de Abreu” within the profile information fields, and 2 of them also contained the string “Brasil”. Several of the profiles contained the string “Casimiro de Abreu” in a way that did not necessarily refer to a location, and at least two of the profiles indicated locations that were outside of Brazil. None of these accounts had recent tweets with geo-location information. As in the Corinto collection, this seed set did not appear to contain many tweets from the target area.
Fixed Profile Energy Function

Attempting to use logistic regression as a profile energy model failed to produce a useful set of users from the target area. The reason for this failure is that there were not enough geo-located users in each iteration inside the target radius to fit a reliable logistic regression model. Poor classifications in each iteration resulted in more collections of users outside the target region in follow-on iterations, and the problem perpetuated.

For this reason we implemented the fixed energy model approach introduced in Section 4.3.2. We used the location specific terms we would have used in list $W_1$ to create three lists:

- $T_1$ A list of character strings or sets of character strings that, if present in a user’s profile information, essentially indicate that a user is in the target location. For example, if a user’s profile contains both of the strings “Casimiro de Abreu” and “Brasil,” we can assume that the user is very likely to be in Casimiro de Abreu, Brazil.

- $T_2$ A list of character strings that, if present in a user’s profile information, strongly suggest that a user is in the target location. “Casimiro de Abreu” is in this list.

- $T_3$ A list of character strings that, if set as a user’s profile location, suggest that a user could be in the target location. “Brasil” is in this list.

These lists are enumerated in Table 4.3. We used these lists to categorize users according to the following algorithm:

1. If a user’s profile meets any of the criteria in list $T_1$, assign category A;
2. Else if a user’s profile location contains a string from the world cities list in Appendix A, excepting Rio de Janeiro, assign category B;
3. Else if a user’s profile location, description, name, or screen name contains a string from list $T_2$, assign category C;
4. Else if a user’s profile location is equal to a string from list $T_3$, or if the profile location is empty, assign category D;
5. Else assign category E.

We then applied the odds table given in Table 4.4 to construct the profile energy function. These odds can be thought of as relationship thresholds required to classify a user in each category into
the target location. Note that users in category C, whose profiles contain keywords or phrases from list $T_2$, are assumed to have a higher likelihood of being outside of the target location. However, a relatively weak relationship with a user inside the target location would be enough to overcome these odds. On the other hand, a user in category B would require many strong connections with users in the target location in order to be classified in the target location.

Application of this odds table to our seed set, without having collected any relationship data, classifies only the two Category A users, whose profiles contain the strings “Casimiro de Abreu” and “Brazil,” in the target location. The remaining nine seed users are initially classified as outside of the target location. Therefore, in the initial expand step only the friends and followers of the two Category A users are queried.

**Table 4.4: Odds Table Used in Naive Implementations**

<table>
<thead>
<tr>
<th>Category</th>
<th>Odds (location:non-location)</th>
<th>$\phi(x_i, 0) - \phi(x_i, 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>50:1</td>
<td>$\log(50)$</td>
</tr>
<tr>
<td>B</td>
<td>1:25</td>
<td>$- \log(25)$</td>
</tr>
<tr>
<td>C</td>
<td>1:2</td>
<td>$- \log(2)$</td>
</tr>
<tr>
<td>D</td>
<td>1:5</td>
<td>$- \log(5)$</td>
</tr>
<tr>
<td>E</td>
<td>1:8</td>
<td>$- \log(8)$</td>
</tr>
</tbody>
</table>

**Performance**

We employed the expand—classify methodology exactly as in the Corinto collection, except that in each iteration we used the fixed odds in Table 4.4 as the profile energy model. After three hours, this implementation collected 99,606 users and had classified 492 of them as being in the target location. Of the 99,606 users in the dataset, 7,729 of them had geo-located tweets, 53 of which were inside the target radius. The resulting AUC for this classifier on the geo-located users was 0.89.
Similar to the Corinto results, the classifier achieves approximately 60% correct detection rate while maintaining a very low false positive rate.

**Sensitivity Analysis**

Figure 4-15 shows the sensitivity of the ROC for this classifier on the geo-located users for several values of $\gamma$ and $\lambda$. The results do not appear to be very sensitive to the value of $\gamma$, but the AUC decreases for lower values of $\lambda$. Larger values of $\lambda$ also seem to show slightly better performance.

![Sensitivity to $\gamma$](image1)

![Sensitivity to $\lambda$](image2)

Figure 4-15: Results and Sensitivity of Casimiro de Abreu user classification.

Note that the performance of the fixed profile energy function as a classifier is also plotted in this Figure for $\gamma = 0$, indicated by a solid black line. The AUC of this classifier is 0.74.

**Summary of Casimiro de Abreu Collection**

The Casimiro de Abreu collection demonstrates the utility of the expand—classify methodology using a relatively naive approach to forming a profile energy function. In the case of Casimiro de Abreu, attempts to implement the same approach using the logistic regression classifier were not successful because there were not enough geo-located users in the initial iterations.

While implementation of this fixed profile energy model is less elegant than fitting a parametric model to labeled data, we found that this naive approach also produced useful classifications in the cases in which logistic regression performed well. Using the same categorization method described in this section and the odds shown in Table 4.4 as the profile energy function in the Corinto factor graph classification, for example, we were able to achieve an AUC as high as 0.87.
4.4.3 Caracas, Venezuela

Caracas, Venezuela, centered at 10.481, 66.904 [64] is the capital of Venezuela. It has a population of approximately 2.1 million. For labeling geotagged tweets, we used a radius of 15 miles from the latitude–longitude coordinates above (see Figure 4-16).

![Figure 4-16: Caracas, Venezuela label radius, plotted on Google Maps][64].

Seed Set

Unlike the previous two locations, querying the Twitter API for Caracas, Venezuela returned 983 user profiles, which is close to the API-imposed maximum of 1000. Of these we used a set of 64 profiles as seed accounts for this collection.

Performance

We ran the expand–classify algorithm, using logistic regression as the profile energy model, for six hours to collect users in Caracas. The set of character strings \( W_1 \) used to extract features from the user profiles is given in Table 4.5. For \( W_2 \) we again used the list of cities in Appendix A, with
Caracas removed. At the end of the six hour period we had 210,656 users in our dataset, 33,261 of which were classified as being in Caracas.

Table 4.5: List of character strings $W_1$ used to extract profile features for Caracas logistic regression.

<table>
<thead>
<tr>
<th>String 1</th>
<th>String 2</th>
<th>String 3</th>
<th>String 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Caracas”</td>
<td>“Caracas, Venezuela”</td>
<td>“Capital, Venezuela”</td>
<td>“caracas, vzla”</td>
</tr>
<tr>
<td>“capital, vzla”</td>
<td>“Caracas Venezuela”</td>
<td>“Capital Venezuela”</td>
<td>“caracas vzla”</td>
</tr>
<tr>
<td>“capital vzla”</td>
<td>“distrito capital”</td>
<td>“venezuela”</td>
<td>“vzla”</td>
</tr>
</tbody>
</table>

In this case our method did not produce the level of performance achieved on the smaller locations discussed in the preceding paragraphs. In fact, the best AUC (0.78) was achieved by simply applying the logistic regression classifier on the profiles; including relationship information does not result in significantly improved performance. Figure 4-17 shows the $\gamma$ sensitivity plot.

![ROC Sensitivity to Link Energy Function Height $\gamma$](image)

Figure 4-17: Caracas, Venezuela Performance.

**Discussion of Caracas Collection**

An obvious difference between Caracas and the other implementations presented is the larger population, and these results give us some indication of the limitations of this implementation method. Some of the challenges associated with building a set of users from a large city are intuitive: collecting and classifying a larger dataset requires more computational resources. However, we found
a less subtle problem apparent in our attempts to collect users from big cities that relates to our modeling assumptions.

**Computational Resources.** The first challenge with collecting user sets from large population centers is the problem of computational resources. Six hours of runtime was not sufficient to collect enough users and links to observe the location homophily implied by our assumptions. Of the 919 users returned by the Twitter user search API and not used as seed users, only 190 appear in our data. This suggests that we have not run enough *expand* steps to discover many of the users in Caracas.

Figure 4-18 shows the computation times for each of the Caracas iterations, plotted as a function of dataset size. The iteration numbers are annotated above the points. These iteration times include the *expand* and *classify* times, but because the dataset size does not significantly affect the collection times in the *expand* step, the increases in iteration times depicted the figure are a result of longer classification optimization times.

![Iteration Times for Caracas](image)

Figure 4-18: Times to complete Caracas collection iterations.

These optimizations were performed using an Intel i3-2120 3.3 GHz processor with 16GB of RAM. Allocation of more computational resources would make collecting user sets from large population centers more tractable. We could also introduce some pruning criteria to reduce the size of
the dataset and speed up the optimization.

**Users in Big Cities.** While the computational challenges of collecting sets of users from large metropolises might be overcome by dedicating more resources, we found other challenges in these collections which might require modifications to the collection method. We have assumed that a user whose profile states he or she is in “Caracas, Venezuela” is generally going to be in Caracas, but for big cities we have found from our geo-located data that this assumption might not be accurate.

Of the 450 users with geo-located tweets in our dataset, only 205 were located within 15 miles of the center of Caracas. The remaining 245 were spread throughout the world. Because of this, a logistic regression classifier using only this feature would classify users whose profile locations are “Caracas, Venezuela” as being outside of Caracas. While we observed this property for some user profiles in each location we collected, only in the largest cities did it appear to adversely affect the results.

Having a preponderance of users whose profiles say they are in Caracas but whose geo-located tweets show they are not brings us to a very important consideration: do we want these users in our target location set? Some might be Caracas residents who are simply traveling, while others could be studying or working abroad. Still others might have lived in Caracas in the past but have permanently moved to another location. Even others could simply be lying.

If we do decide these users should be in our dataset, then our approach to fitting a logistic regression model on geo-located users needs to be reworked, because this method of labeling is clearly not a valid proxy for our target set. If, on the other hand, we do not want these users in the dataset, our method of using the geo-located tweets remains valid, but it brings us to another big city challenge to our assumptions: homophily.

While we might not know why a user would have a profile location of “Caracas, Venezuela,” but tweets geo-located elsewhere in the world, we have observed that many of these users tend to have close connections with other users that appear to be in or near Caracas. These high-energy, long-distance relationships run counter to our assumption that close relationships tend to indicate shared location. One plausible conjecture is that people in big cities tend to be more mobile than people from smaller towns, and that mobility has resulted in a larger number of long-distance social media relationships with high link energy scores. This big-city phenomenon was also observed and documented by Backstrom et al. [9]. As a result of it, the homophily that proved useful in the Corinto user classifications is more difficult to exploit in Caracas.
4.4.4 Summary of Results on All Locations

We collected user datasets from a total of nine locations of varying size and culture. Table 4.6 summarizes the results of these collections. We executed all of the collections for 3-6 hours. We note that in general, the optimal observed value for $\gamma$ tends to be lower for locations with larger populations. As we found in our Caracas user collection, there are two likely reasons for this. First, we did not devote enough computational resources and time to collect and classify the larger set of users. Second, location-based homophily is less evident in larger population centers, and therefore more difficult to exploit (see Backstrom et al. [9]).

Even in small population centers, we find that many of the misclassified geo-located users are those whose profiles and connections indicate they belong in the target location, but whose geotagged tweets fall outside of the target location. This qualitative observation indicates that our method of finding users associated with specific locations is, in some cases, performing better than our evaluation criteria suggest.

<table>
<thead>
<tr>
<th>Location</th>
<th>Approx. Population</th>
<th>LR AUC</th>
<th>Model AUC</th>
<th>Best $\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corinto, Colombia</td>
<td>30,000</td>
<td>0.75</td>
<td>0.92</td>
<td>log(10)</td>
</tr>
<tr>
<td>Popayán, Colombia</td>
<td>440,000</td>
<td>0.84</td>
<td>0.88</td>
<td>log(2)</td>
</tr>
<tr>
<td>Greater Binghamton, NY</td>
<td>250,000</td>
<td>0.87</td>
<td>0.90</td>
<td>log(2)</td>
</tr>
<tr>
<td>Casimiro de Abreu, Brazil</td>
<td>35,000</td>
<td>0.74†</td>
<td>0.84</td>
<td>log(8)</td>
</tr>
<tr>
<td>Caracas, Venezuela</td>
<td>2,100,000</td>
<td>0.78</td>
<td>0.78</td>
<td>0</td>
</tr>
<tr>
<td>San Fernando, Venezuela</td>
<td>165,000</td>
<td>0.82</td>
<td>0.9</td>
<td>log(8)</td>
</tr>
<tr>
<td>El Vigía, Venezuela</td>
<td>156,000</td>
<td>0.77</td>
<td>0.90</td>
<td>log(6)</td>
</tr>
<tr>
<td>Zamboanga City, Philippines</td>
<td>19,542</td>
<td>0.81</td>
<td>0.87</td>
<td>log(2)</td>
</tr>
<tr>
<td>Asunción, Paraguay</td>
<td>2,200,000</td>
<td>0.64</td>
<td>0.7</td>
<td>log(2)</td>
</tr>
</tbody>
</table>

† AUC from fixed profile energy model.

Using Twitter’s native methods for finding users in or associated with a specific location reveals the same set of challenges. In our discussion of the Corinto collection results, we briefly compared our user dataset to those obtained using two of Twitter’s search APIs. We now provide a more general discussion of the performance of these methods compared with our results.
**Twitter User Search**

The Twitter user search API enables a person to search for Twitter users based on a query string [140]. The API returns user profiles that contain a match or partial match of the query string in the profile information. We used this method to produce the seed sets of users for all of our collections. This method can return up to 1000 profiles for a specific search query.

In several cases, this API did not return any results for specific location queries. These queries include “Corinto, Colombia,” “Casimiro de Abreu, Brazil,” and “San Fernando de Apure, Venezuela.” In these cases we used the city or town name only in the query to produce results, with more unique town names producing better seed sets. The seed set from the “Casimiro de Abreu” query included only two accounts that appeared to be in or related to the target location. The other user profiles in this seed set mentioned “Casimiro de Abreu” in another context, such as a similarly named location in another country, or even as a person’s name.

Even when the specific location query returned results, the profiles obtained were not necessarily in the target location. Among the results from the “Binghamton, NY” query, for example, was a user from Virginia whose only apparent connection to Binghamton was a claim in his profile description that he had once met his favorite celebrity there. This was the only profile returned by this user search query that did not appear in our Binghamton collection.

Using the user search for Caracas returned close to the maximum of 1000 users. We found geo-located tweets for 102 of these users. Of these, only 27 were inside of our 15 mile radius around Caracas (Figure 4-16), while the remaining 75 were scattered around the world.

**Twitter Search**

The Twitter search API returns tweets that contain a match or partial match for a query string. This API also takes an optional location and radius, and returns only matching tweets originating from inside the radius. According to the documentation, the API uses tweet geotagging if available, but otherwise will use profile location information Twitter [139].

One of the primary drawbacks of attempting to use this method to build a set of users is the need to supply a query string, as this API searches for tweets and not users. Also, the Twitter search API limits its results to tweets from the previous week, so less active users would not be found using this method.

We attempted to employ this method by executing a Twitter search query for the single character
e, and supplying the latitude–longitude location and radius that was used for labeling geotagged
tweets in each collection.

The users that posted the tweets returned by these queries did not generally appear to be in the
corresponding target locations. For example, executing this query using the location of the greater
Binghamton area returned tweets from 66 users. Of these profiles, only 6 indicated they were in
the greater Binghamton area. Five of these users appeared in our dataset and 4 were classified as
being in the target location.

Many of the remaining 60 user profiles returned by the search query indicated user locations that
were clearly not in the target area. The most frequent location was Maine; 20 of the users indicated
they were in this state in their location fields. Others gave locations throughout the United States,
and a few were located in Canada, South America, or Europe.

Searching through the 66 user timelines for geo-located tweets only yielded two locations: one in-
side the Binghamton area and one outside. The one user that was geo-located inside the Binghamton
area was also one of the six users that indicated he was in Binghamton in his profile location.

Because many users obtained through the Twitter search API for each location indicated in their
profile that they were not actually located in the target area, we concluded that using the Twitter
search API to obtain users from a specific location is not reliable.

4.5 Non-location Grouping

In order to evaluate this effort to a non-location target group, we applied it in attempt to collect
a specific ideological grouping of users. The ideological grouping we chose was the group of users
supporting and propagating “Pizzagate,” which is an extreme and unsubstantiated allegation of mis-
conduct against members of the Hillary Clinton 2016 presidential campaign [122]. We qualitatively
observed that users that appeared to believe in this conspiracy theory tended to follow and interact
with each other in Twitter (this might be an example of an “echo chamber” as described by Pentland
[113]).

Because we lacked labeled data, we used a very simple model to build our profile energy function.
If a user mentioned “pizzagate” in his or her profile information, we assumed the user was 200 times
more likely to be a believer in the pizzagate ideology. If a user did not mention “pizzagate,” we
assumed the user was 20 times more likely to not believe in the pizzagate ideology. We used the
same link potential function as in the location user collections.
After six hours of collection we had 289,095 users in the dataset, 12,784 of which were classified as pizzagate subscribers. Only 433 of these users mentioned “pizzagate” in their profiles. We then hand-labeled a random sample of 195 users from this dataset, 97 of which were classified as pizzagate believers by the algorithm. The order of the labeling was randomized and the labeler did not have access to the classifications, making his judgments based solely on manual inspection of each user’s profile and tweets. The hand-labeling of points was naturally subjective; in most cases there was no way the labeler could know for sure whether a user subscribed to the Pizzagate allegations.

Using these labels as a test set produced the ROC curve depicted in Figure 4-19. We find that the algorithm performs well in predicting these human-generated labels.

4.6 Conclusion & Future Research

Obtaining a set of social media users from a specific location is a difficult problem [41]. Consistently reliable ground truth data does not exist. We can benefit from using geo-located content, but this data is only available for a small fraction of users and is also subject to uncertainty, as users do not necessarily post geo-located content from their homes. More widely available data, such as that contained in user profile information, can be even less reliable.
Evaluations of our own method show that it provides a useful collection and classification tool for population centers with up to about 500,000 people. The ROC curves generally show effective location classification. When compared with other methods of collecting location users, our method produces more results and, again excepting big cities, does not appear to miss many of the users in the target location that are captured by the other methods. Again we note that this final comparison is difficult because other methods do not consistently produce reliable results.

Adapting the expand—classify methodology to effectively collect sets of users from large population centers is an area that deserves further inquiry. Dedication of more computational resources and time might immediately yield useful results. It is also possible that some refinements to our implementations will be necessary to improve performance on big cities. These improvements could address the increase in intercity mobility, observed in large cities [9], which results in less reliable geo-location data and user profile information.

While the objective of this effort has been to produce a reliable and somewhat comprehensive set of social media users from a specific location, as mentioned at the beginning of this chapter this method could be applied to any social media grouping which exhibits some level of homophily and exhibits some indicators of group membership through users’ profile features. The role of the factor graph model we utilize is to provide a means of considering both the indicators present in a user’s profile and the user’s social network connections when making classifications. In our implementations, we adjusted the parameter $\gamma$ to control for the amount of influence a user’s social network connections had in his or her classification.

Fitting a reliable parametric profile energy model on labeled data was feasible in most cases, even though the location labels obtained from geo-located content were not perfect. If a similar method existed to label relationships, so that for some subset of relationships we knew whether or not the connected users shared the same location, then it might also be possible to fit a parametric model for the link energy function as well.

We have found that fitting parametric models on the energy functions based only on the classification labels coming from the Energy Graph optimization does not always lead to favorable results. Zabih and Kolmogorov [151] suggest this type of Expectation-Maximization approach in removing noise from images, but in this application we have observed that it often leads to an undesirable local minimum, with all or almost all users receiving the same classification.

Finally, this method might be improved by prioritizing friend and follower queries in the expand step. Some users might be more inclined to have local friends and followers. If a probability model
can be established that quantifies these inclinations, the network search methods of Alpern and Lidbetter [5] or our work in Chapter 2 might be leveraged to grow the set of users in the target location more efficiently.
Chapter 5

Conclusion

In this thesis, we have presented and evaluated several useful network search methodologies and their applications, applying the methods of operations research and machine learning to investigate how social media and social network data can be used to improve situational awareness from a military or security perspective. In Chapter 2 we presented a multi-urn search problem as a model for searching for a specific vertex in a network. Using this model, we proved that there is always an optimal block policy in searches that meet the multi-urn search problem assumptions, irrespective of correlations in the probability model. We also provided necessary and sufficient conditions for block policy optimality in two specific cases: independent urns and the single red marble scenario. Finally, we gave a few properties of the dynamics of the multi-urn search problem and commented on the challenges of finding more general optimality conditions.

In Chapter 3, we combined statistical modeling of extremist behavior with our optimized search policies from Chapter 2. Our behavioral modeling allowed us to predict new extremist users, determine if two accounts belong to the same extremist user, and predict the network connections of suspended extremist users when they create new accounts. We used our behavioral models to formulate a network search policy to find the new accounts of suspended extremist users when they return to the social network. Simulations based on actual ISIS users found that our policy was much more efficient than other benchmark approaches.

Finally, Chapter 4 addressed the problem of obtaining a set of social media users from a specific location. We developed and implemented an expand—classify methodology that employed a factor graph model and, in cases in which labeled data could be obtained, logistic regression. We demonstrated the performance of this method on several locations, finding that our method outperformed
Twitter’s native search methods. Evaluations of our expand—classify implementations show that it provides a useful collection and classification tool for population centers with up to about 500,000 people. We found that obtaining the set of users from a large population center was particularly challenging, and we enumerated and discussed how some of these challenges might be overcome.
Appendix A

World Cities Data

The following cities were used in set $W_2$ in our specific location collection implementations. This list is extracted from the World Cities Dataset created and maintained by MaxMind, available at http://www.maxmind.com/ [99]

dubai      kabul    yerevan   luanda   cordoba   rosario
vienna    adenide   brisbane  melbourne   perth       sydney
baku      dhaka     khsna     brussels    ouagadougou    sofia
belem     beo horizonte manaus   campinas    curitiba    fortaleza
rio de janeiro salvador sao paulo   nova iguaçu porto alegre recife
vancouver   kinshasa  hubumbashi brazzaville montreal   toronto
ouala      yaounde   anshan    changwun   abidjan    santiago
dalian     datong    fushun    fuzhou     guangzhou   guiyang
handan     hangzhou  harbin    hefei      huainan   jilin
jinan      kunning   lanzhou   luoyang    nanchang   nanjing
peking     qingdao   rongchung shanghai   shenyang   shenzhen
suzhou     taiyuan   tangshan  tianjin    urumqi      wuhan
wuxi       xian      xianyang  xinyang    xuzhou     barranquilla
bogota     cali       medellin    prague   berlin     hamburg
munch      copenhagen santo domingo algiers     guayaquil   quito
alexandria cairo      gizheh      barcelona madrid     addis ababa
paris      london     tbilisi    accra      kumasi     conakry
guyport     budapest  bandung    bekasi     depok      jakarta
maka    masar    palembang   semarang   surabaya   tangerang
dublin    agra     ahmadabad   allahabad   amritsar   aurangabad
bangalore  bhopal   bombay    calcutta    delhi       faridabad
<table>
<thead>
<tr>
<th>ghaziabad</th>
<th>haora</th>
<th>hyderabad</th>
<th>indore</th>
<th>jabalpur</th>
<th>jaipur</th>
</tr>
</thead>
<tbody>
<tr>
<td>kalyan</td>
<td>kanpur</td>
<td>lakhnau</td>
<td>ludhiana</td>
<td>madras</td>
<td>nagpur</td>
</tr>
<tr>
<td>new delhi</td>
<td>patna</td>
<td>pimpi</td>
<td>pune</td>
<td>rajkot</td>
<td>surat</td>
</tr>
<tr>
<td>thana</td>
<td>vadodara</td>
<td>varanasi</td>
<td>visakhapatnam</td>
<td>baghdad</td>
<td>esfahan</td>
</tr>
<tr>
<td>karaj</td>
<td>mashhad</td>
<td>qom</td>
<td>shiraz</td>
<td>tabriz</td>
<td>milan</td>
</tr>
<tr>
<td>rome</td>
<td>hiroshima</td>
<td>kawasaki</td>
<td>kobe</td>
<td>nagoya</td>
<td>saitama</td>
</tr>
<tr>
<td>tokyo</td>
<td>nairobi</td>
<td>phnom penh</td>
<td>seoul</td>
<td>almaty</td>
<td>bayrut</td>
</tr>
<tr>
<td>beirut</td>
<td>tripoli</td>
<td>casablanca</td>
<td>fez</td>
<td>rabat</td>
<td>antananarivo</td>
</tr>
<tr>
<td>bamako</td>
<td>mandalay</td>
<td>rangoon</td>
<td>ecatepec</td>
<td>guadalajara</td>
<td>juarez</td>
</tr>
<tr>
<td>leon</td>
<td>mexico</td>
<td>monterrey</td>
<td>nezahualcoyotl</td>
<td>puebla</td>
<td>tijuana</td>
</tr>
<tr>
<td>kuala lumpur</td>
<td>maputo</td>
<td>benin</td>
<td>ibadan</td>
<td>kaduna</td>
<td>kano</td>
</tr>
<tr>
<td>lagos</td>
<td>maituguri</td>
<td>port harcourt</td>
<td>managua</td>
<td>lima</td>
<td>davao</td>
</tr>
<tr>
<td>manila</td>
<td>faisalabad</td>
<td>gujaranwala</td>
<td>hyderabad</td>
<td>karachi</td>
<td>lahore</td>
</tr>
<tr>
<td>multan</td>
<td>peshawar</td>
<td>rawalpindi</td>
<td>warsaw</td>
<td>bucharest</td>
<td>belgrade</td>
</tr>
<tr>
<td>chelyabinsk</td>
<td>kazan</td>
<td>moscow</td>
<td>nizhniy novgorod</td>
<td>novosibirsk</td>
<td>omsk</td>
</tr>
<tr>
<td>rostov-na-donu</td>
<td>saint petersburg</td>
<td>samara</td>
<td>ufa</td>
<td>volgograd</td>
<td>yekaterinburg</td>
</tr>
<tr>
<td>jiddah</td>
<td>mecca</td>
<td>riyadh</td>
<td>khartoum</td>
<td>aleppo</td>
<td>damascus</td>
</tr>
<tr>
<td>singapore</td>
<td>freetown</td>
<td>dakar</td>
<td>mogadishu</td>
<td>gaziantep</td>
<td>istanbul</td>
</tr>
<tr>
<td>bangkok</td>
<td>adana</td>
<td>ankara</td>
<td>bursa</td>
<td>taipei</td>
<td>dar es salaam</td>
</tr>
<tr>
<td>izmir</td>
<td>kaohsiung</td>
<td>kaohsiung</td>
<td>taichung</td>
<td>los angeles</td>
<td>san diego</td>
</tr>
<tr>
<td>kiev</td>
<td>odesa</td>
<td>kampala</td>
<td>phoenix</td>
<td>houston</td>
<td>san antonio</td>
</tr>
<tr>
<td>chilango</td>
<td>new york</td>
<td>philadelphia</td>
<td>dallas</td>
<td>valencia</td>
<td>hanoi</td>
</tr>
<tr>
<td>montevideo</td>
<td>tashkent</td>
<td>caracas</td>
<td>maracaibo</td>
<td>johannesburg</td>
<td>pretoria</td>
</tr>
</tbody>
</table>


176


[65] Cat Graham and Chris Thompson. A guide to social media emergency management analytics; understanding the place of analytics through typhoon haiyan social media analysis. Published by Humanity Road and Statistics Without Boarders, November 2014.


183


