DYNAMICS AND STABILITY OF EDGES AND COMMUNITIES IN SOCIAL NETWORKS

Amr Elsisy

Submitted in Partial Fullfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

Approved by:
Dr. Boleslaw K. Szymanski, Chair
Dr. Malik Magdon-Ismail
Dr. Jianxi Gao
Dr. John Mitchell

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York

[December 2021]
Submitted November 2021

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

This work was supported by the U.S. Department of Homeland Security under Grant Award Number 2017-ST061-CINA01, the Defense Advanced Research Projects Agency (DARPA) under Contract No. W911NF-17-C-0099, the Army Research Office (ARO) under Contract No. W911NF-16-1-0524 and by the Army Research Laboratory under cooperative agreement W911NF-09-2-0053 (NS-CTA). The research and conclusions drawn in this thesis are those of the authors and are not necessarily representing the official policies of the funding agencies.

I would like to thank everyone that has contributed towards this thesis. First and foremost, I would like to thank my advisor, Professor Boleslaw K. Szymanski for supporting me throughout my PhD. Professor Szymanski has always pushed me to learn, and has always been there to guide me through any difficulties I had, whether they were work related or not. I would also like to thank Professor Thomas Sharkey for always providing research related advice during the early phase of my PhD. Finally I would like to thank my committee members, Professor Malik Magdon-Ismail, Professor Jianxi Gao, and Professor John Mitchell for helping me realize several good research ideas that have shaped the research I have done during my PhD.

Further I would like to thank my lab mates, Ashwin, James, Aamir, and Yeming, for providing a great lab environment, and for always being there to discuss the research I am conducting, and helping me think of new ways to approach any sort of obstacles I was facing. I would also like to thank all my friends in Troy for helping make graduate school more fun than it normally is. Thanks to Michael, Bariscan, Andres, Javad, Shailesh, Omar, Ali, Stavros, George, Marcelo, and Maha, I was able to take much needed breaks from work, which helped me clear my head and recharge.

Finally, I would like to thank my friends and family from back home for supporting me from afar. First, I would like to thank my father, Elnady Elsisy for always being the support system I needed, and for always lifting me up whenever I needed it. Thanks to him I continued my graduate school career, and if it was not for him, I would not be where I am today. I will forever be grateful to him. I would also like to thank my sister, Essraa Elsisy for her constant calls and for always being there for me. Her calls were always a breath of fresh air, and always helped me refocus. Also to my aunt Hala Elsisy, I am thankful for how caring, and loving you are, and for always being supportive. I would also like to thank Ahmed Elhefny for constantly travelling to the US, and for always encouraging me to take much needed breaks, which ultimately helped recharge me, and

made me much more productive afterwards. I would also like to especially thank him for flying all the way to the US to attend my PhD defense. Lastly, I would like to thank Farah Al Barqouni for always being there whenever I needed her. Farah has been with me during the lowest times of my graduate school career, and her words of encouragement and constant support have helped me in ways that I cannot describe. Thank you Farah, for always being there, and especially for always allowing me to repeatedly practice my talks with you. Farah has given me the most support during my PhD career, despite how far away she is. I am forever grateful to my friends and family, and therefore, I dedicate this thesis to all of them.

ABSTRACT

We study social and criminal networks, with a focus on how users and communities of such networks interact with one another, and the impact that such interactions have on the community structures of such networks. We first conducted a study on the crime rates present in different community areas across the city of Chicago, and we were able to identify and predict which community areas have the highest crime rates. In this study, we did not have information about how these community areas interacted with one another. This led us into the study of the Gowalla network, where we analyzed how members of the Gowalla network interacted with one another, and how the modifications of such interactions can break the structure of the original Gowalla network. Such modifications involve the addition and removal of edges from the network, which then led us into our third study, which was creating a synthetic network generator to rewire, with the extent defined by a parameter, edges of the given real-world network. When repeated, this rewiring process adds and removes edges from many generated synthetic versions of the original network. Consequently, all generated networks are also statistically similar to each other. We found that the networks generated using this approach often had community structures different from the one that the original network had. This led us into our final study of measuring the entropy and uncertainty present in the generated networks' community structures. Repeating rewiring enables us to identify the generated network with the lowest community structure uncertainty. Such a network and its corresponding community structure can be used as the best rewired version of ground truth structure alternative to the original network. Finally, we found that predicting the community structure with the lowest cost of this network uncertainty lowers not only this cost but also uncertainty itself. The cost function can be defined according to the requirements of the applications.

CHAPTER 1
INTRODUCTION

We study the properties and structures of social networks, with a focus on covert networks. We start off with studying criminal activities within a network of city community areas. We present our work done on predicting crime in the city of Chicago, which is broken down into 77 community areas. We introduce a method for predicting crime in community areas, using the census data, and past crime data available for the community areas. We show the process of eliminating features that were not helpful to our prediction method, and rank the most useful features. We find that features representing demographic characteristics were not as important as features representing crime rate. We develop a predictive algorithm to forecast future crime rates in each community area in a given year, using past crime rates, and past neighborhood crime rates for a given community. We then further study the community areas to better understand which of them would yield the largest city wide crime reduction, if crime reduction funding was invested in them. We find that investment of crime reduction funding in different community areas may result in vastly different overall crime reductions, thus showing that any interventions need to be carefully planned.

We then study a real social network, Gowalla, which is a location-based activities network that allows its users to ”check in” at physical location, and share this information with their friends. Using the geographic coordinates, and friendships of the users in the Gowalla network, we emulated Milgram’s small world social search experiment. We then shuffled the geographical coordinates of the users, and the friendships relations of the users, using several different distributions, to find whether these distributions impact the social search efficiency. We find that the embedding of nodes into space has no effect on efficiency of the social search, and that only the spatial distribution of friends affects the social search efficiency. The actual friendship edges did not matter, as long as the spatial distribution of friends was similar to the original spatial distribution of friends in the Gowalla network. We also found that partial knowledge of friends of friends improves social search.

We proceeded with creating a social network generator, with a focus on covert networks, such as terrorist or criminal networks. Unlike the members of open social networks, members of covert networks attempt to hide their membership and activity, and thus data about covert networks tends to be incomplete and/or inaccurate. We propose a Bernoulli Weighted Random Network (BWRN) generator to address noisiness of covert network’s data, and to deal with the uncertainty regarding

the completeness of information in a given covert network. The BWRN generator collects statistical information about the network's edges, based on the communities and leadership hierarchy present in the network. The BWRN generator will then create synthetic networks with the same statistical properties as the original network possesses. The generated networks can then be used to measure the stability of the original network, by analyzing the stability of community structures across the different generated networks.

Finally, we proceed with using the BWRN generated networks to measure the uncertainty of the provided real network, especially when the provided network has noisy data. We look into the application of the BWRN generator to covert networks, which have noisy data, arising from imperfect data collection methods. We study the generated networks' community structures to help shed some light on the stability and uncertainty of the original network. Using the BWRN generator we rewire a real-world network, thus generating a universe of networks with varying edge structures, and for each of those networks we detect its community structure. We then use two entropy-based metrics to measure the uncertainty present across the community structures of the generated networks, and identify the network with the lowest community structure uncertainty. The first proposed entropy-based metric is the Shannon entropy-based metric, which is used to measure the uncertainty present in the generated networks' community structures. The second proposed entropy-based metric is the *set* entropy based metric, which is used to measure the uncertainty, and limits of predictability, on the entire set of generated networks' community structures. Using such entropy-based metrics we are able to quantify the uncertainty present in the generated network's community structures. We also propose a heuristic that builds the community structure that has the lowest expected cost of uncertainty, and such cost can be individualized to the needs of different applications.

CHAPTER 2
CRIMINAL ACTIVITIES IN CITY COMMUNITY AREAS

The contents of this chapter are based on the work published in [1], and contains the parts of the publication that were originally contributed by the author.

2.1 Overview

The work presented focuses on criminal activities and patterns of crime in the city of Chicago. The city of Chicago is broken down into community areas, and for each community area we have census data, and criminal activities data, that were collected over 16 consecutive years. Using this unique dataset, we study the spatio-temporal distribution of crime, and find that there is a presence of criminal hot-spots. The presence of criminal hot-spots helps us better understand the nature of crime activities. We develop predictive algorithms that incorporate both census data, and crime rates of community areas, to forecast the number of future crimes in city community areas. Lastly, each community area is examined in order to understand which area would result in the largest reduction of crime, when considering the increase of crime reduction funding. The community area which yields the largest city wide crime reduction, is considered to be an optimal community area to invest such funding.

2.2 Introduction

Criminal activities have been extensively studied over the years in an effort to try and reduce crime [2]–[4]. Crime reduction and prevention requires us to have a deep understanding of crime patterns and the dynamics of crime, which is not easy to grasp due to the complex nature of criminal activities [5]. The considerable data available nowadays, and the progression of computational tools, permits researchers and police forces to gain further knowledge on the dynamic nature of crimes. This will ultimately improve the detection and prevention of criminal activities. Criminal records have recently become publicly available by many cities across the U.S., which can be of great help to researchers. Researchers can use such data to forecast crime, and to ultimately work hand in hand with the police force to prevent future crimes. Understanding the nature of crime

Portions of this chapter previously appeared as: X. Niu, A. Elsisy, N. Derzsy, and B. K. Szymanski, "Dynamics of crime activities in the network of city community areas," Appl. Netw. Sci., vol. 4, no. 1, p. 127, Dec. 2019.

can help us optimize the allocation of funding in a way that yields the greatest overall reduction in crime.

The existence of criminal hot-spots has been shown through many studies that were performed on government released data regarding crime activities in cities [6]–[10]. To build our predictive model we take into account the spatio-temporal patterns of crime activities in cities. We also take into account the census data of such community areas, to find if they have any impact on our prediction accuracy. The final contribution of this chapter is the proposal of a crime reduction optimizer, which studies how extra funding impacts community areas individually. Each community area is studied to find how much impact it has on the reduction of overall crime in the city. The community area with the highest impact on crime reduction is then proposed as the optimal community area for investment of the crime reduction funding.

2.3 Data

The data which we collected and analyzed contains reports on crimes that occurred in the city of Chicago. Since the city of Chicago is divided into 77 community areas. Crime records, and census data were allocated for such communities. The crime records were available for 16 consecutive years (2002 to 2017), and the census data for 18 consecutive years (2000 to 2017). It is important to highlight that while the crime data was available on a yearly basis, the census data was not. The census data was only available for the years 2000, 2008, 2012, 2013, and 2017. Thus, we had to interpolate this data from the reported years, to obtain yearly estimates of census data.

Demographic data was extracted from the census data provided by the U.S. Census Bureau (U.S. Census Bureau). Crime information was extracted from the crime incident reports obtained from the City of Chicago Data Portal, the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system, and the IUCR (Illinois Uniform Crime Reporting) codes (Chicago Police Department). We limit the data used in this study to records of violent crimes (i.e., burglary, assault, homicide) as identified by IUCR.

2.4 Distribution of Criminal Activities

At the first glance of this data, we find that the distribution of crime across community areas is not uniform. As foreseen, some community areas had high crime rates, while others had very low crime rates. In Fig. 2.1, we show a histogram of the number of crimes occurring across the 77

community areas for the year 2017. We find that between the community areas with the highest rate of crime, and the community areas with the lowest rate of crime, there is a difference of about 15 fold in the number of crime incidents.

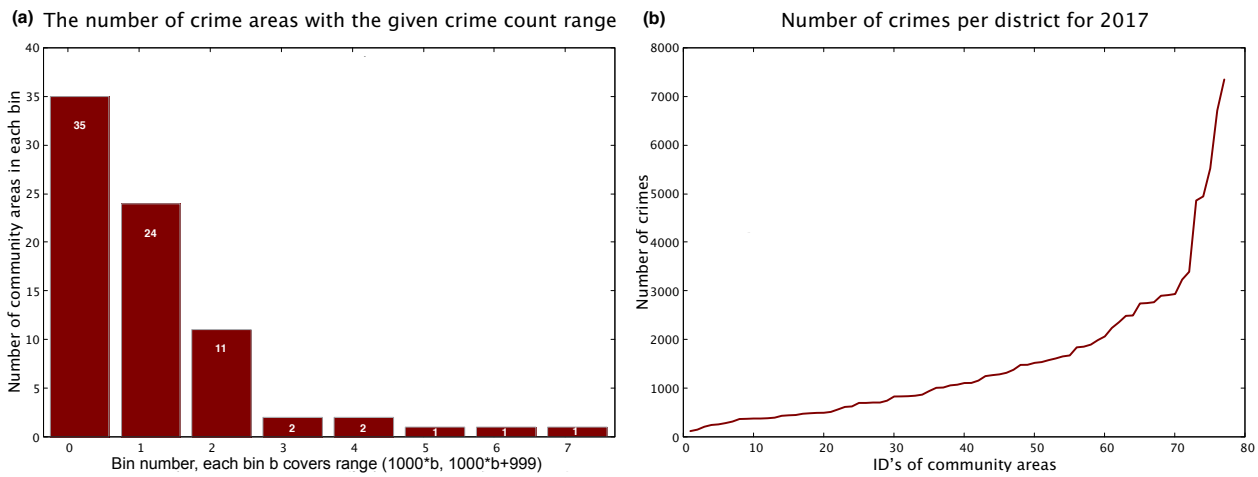

Figure 2.1: (a) Histogram showing the crime distribution across community areas with varying numbers of crime occurrences. Ranges for bin $b = 0, 1 \ldots, 6, 7$ is $[b * 1000, b * 1000 + 999]$. (b) Community areas are listed in the increasing order of their recorded number of crime incidents in the year 2017.

As seen in Fig. 2.1, the majority of crimes that were occurring in the city of Chicago, for the year 2017, were concentrated in just a few community areas. We studied the crime rates of the 77 community areas over the 16 years of data that we have. We find that the communities with the highest crime rates are more or less consistent all throughout the 16 years (2002 to 2017), see Table 2.1. We see that communities 37, 32, and 40, are consistently ranking in the top 3 communities with the highest crime rates. We can see that throughout the 16 years, the highest crime rate community is either community 37 or community 32. Meanwhile, the 5th position has a lot more variations, where six different communities (26, 28, 40, 44, 67, 68) are placed in that position throughout the years.

2.5 Predicting Yearly Crime Rate

Our goal here is to use the provided data, to better understand the patterns of crime, and to be able to accurately predict future criminal activities. This will help us gain a thorough understanding of which census data, or past criminal data, can be utilized to better forecast future crimes, which can help us prepare for the future, and act as an instrument for crime prevention to mitigate future

Table 2.1: For all years of data, we look at communities with crime rates that exceed the average crime rate by at least 1 and 1/2 of the standard deviation for that year.

year	1st		2nd		3rd		4th		5th		6th	
	Id	rate	id	rate	Id	rate	Id	rate	Id	rate	Id	rate
2002	37	0.158	32	0.142	40	0.111	68	0.107	38	0.097		
2003	37	0.184	32	0.177	40	0.147	44	0.132	68	0.128	67	0.126
2004	32	0.185	37	0.180	40	0.143	68	0.129	44	0.125	69	0.121
2005	32	0.175	37	0.174	40	0.143	68	0.127	44	0.122	67	0.120
2006	37	0.175	32	0.175	40	0.152	68	0.132	67	0.122	26	0.121
2007	32	0.177	68	0.142	40	0.140	37	0.140	67	0.128		
2008	32	0.187	37	0.173	68	0.145	67	0.129	40	0.120		
2009	32	0.168	37	0.135	68	0.133	40	0.120	67	0.117	69	0.106
2010	32	0.161	37	0.158	68	0.129	40	0.122	67	0.117		
2011	37	0.190	32	0.153	40	0.124	68	0.122	67	0.116		
2012	37	0.171	32	0.154	40	0.114	68	0.113	26	0.106	67	0.102
2013	32	0.142	37	0.137	40	0.110	68	0.100	67	0.095	44	0.090
2014	32	0.134	37	0.127	40	0.095	68	0.091	44	0.082		
2015	32	0.144	37	0.130	40	0.098	68	0.084				
2016	37	0.172	32	0.163	26	0.096	40	0.094				
2017	32	0.187	37	0.170	26	0.098	40	0.094				

crimes.

2.5.1 Methodology

The spatio-temporal information available from the census, and criminal data that we collected were the following. For each community area we had *CommunityID*, *Date*, *NumberOfIncidents*. The demographic data corresponding to each community area were: *HomeOwnershipRate*, *PovertyRate*, *HighschoolDegreeRate*, *BachelorDegreeRate*. With such data available, we used a spatio-temporal linear regression algorithm [11] to forecast yearly crime rates for a given community area. The implementation of the used algorithm is as follows. Let $a_{i,t}$ be the number of crime incidents at community i during year t, and $v_{i,j,t}$ be the value of a feature j during year t and at community i, with $v_{i,1,t} = a_{i,t}$. In order to predict the number of crime incidents for some future year t at each community i we use a vector $\vec{y}_t = \{a_{1,t}, a_{2,t}, ..., a_{i,t}, \ldots, a_{77,t}\} \in R^L$, and the input matrix $X_t \in R^{L\times\tau\times n}$, where $L = 77$ denotes the total number of communities, n is the number of features, and τ is the length of the historical data used. Each feature value is defined at certain community area i and historical time (year) t. Hence, each value of the input matrix at current time of prediction t_p is defined as $X_{t_p} = (v_{i,j,t})$, $i \in [1,L]$, $j \in [1,n]$, $t \in [t_p - \tau + 1, t_p]$. In the

spatio-temporal linear regression model [11], at current time t_p we have

$$\vec{y}_{t_p+1} = wX_{t_p}. \tag{2.1}$$

To obtain the optimal matrix w, we use regression on known past data about crime and features values using the following equation

$$w = \underset{w}{\operatorname{argmin}} ||\vec{y}_{t_p} - wX_{t_p-1}||^2_{1+\tau\times n}. \tag{2.2}$$

2.5.2 Feature Selection and Forecasting Results

For yearly crime prediction, we focus on per capita crime, as it is a more accurate reflection of which community areas have higher crime rates. Per capita crime accounts for the population of community areas, which is available as part of the census data. Our first task is to pick the set of features that we will use for our prediction algorithm. From the mentioned available demographic, and crime data, we chose the following set of features. $F1$ *HomeOwnershipRate*; $F2$ *PovertyRate*; $F3$ education level, which is a combination of both *HighschoolDegreeRate* and *BachelorDegreeRate*; $F4$ *NeighborhoodCrimeRate*, which is the average crime rate over the neighboring communities (sharing a border) to a given community, and the last feature $F5$ *CommunityCrimeRate*.

To determine which of those features are of importance to our prediction accuracy, we ran the null hypothesis test. We start off with all the mentioned features from the year previous to our prediction year. We run the null hypothesis test, and remove those features that do not pass the null hypothesis test. To pass the null hypothesis test, a feature requires a P-value less than or equal to 0.05. As shown in Table 2.2, feature $F5_1$, is the only feature that survives with a P-values less than 0.05. We also noticed that feature $F4_1$ was not far off, and therefore we keep it, to test it even further. Note, the index of each feature indicates the number of years before the year of prediction, in which the value of the feature is computed. With feature $F5_1$ we get an R-squared measure of 0.9338. Moving forward, we add features for the second year before the forecast year, and only preserve the features that pass the null hypothesis test. Our set of possible features is: $F5_1, F4_1, F5_2$, and $F4_2$, of which all pass except for $F5_2$. With $F5_1, F4_1, F4_2$ we get an R-squared measure of 0.9635. We repeat this process for a third year before the forecast year, but find that no improvements are observed, so we decide to keep our set of features of $F5_1, F4_1, F4_2$, as our optimal set of features.

The results achieved using the features $F5_1, F4_1, F4_2$, giving us an R-squared measure of

Table 2.2: Null hypothesis test to determine which features, with a current length of historical data, are optimal to use for crime forecasting. $F5_1, F4_1, F4_2$ are the optimal set of features to use.

Historical data depth	Coefficients	P-values	Coefficients	P-values	R-squared metric
1	$F1_1 = -0.0017$ $F2_1 = -0.0016$ $F3_1 = -0.0037$ $F4_1 = -0.0888$ $F5_1 = 1.2198$	*0.6358* *0.6087* *0.5905* *0.0631* $< 10^{-47}$	$F5_1 = 1.1786$	$< 10^{-55}$	0.93
2	$F4_1 = 0.7661$ $F4_2 = -0.8118$ $F5_1 = 1.2432$ $F5_2 = -0.1234$	0.0002 $< 10^{-5}$ $< 10^{-20}$ *0.2738*	$F4_1 = 0.8484$ $F4_2 = -0.8955$ $F5_1 = 1.1181$	$< 10^{-5}$ $< 10^{-6}$ $< 10^{-89}$	0.96
3	$F4_1 = 1.1177$ $F4_2 = -1.0378$ $F4_3 = -0.0702$ $F5_1 = 1.1967$ $F5_2 = -0.2443$ $F5_3 = 0.0953$	$< 10^{-9}$ $< 10^{-6}$ *0.6889* $< 10^{-28}$ *0.0334* 0.2526			

0.9635, demonstrates the following. The demographic characteristics of community areas are so-well encoded in the features used, that the direct use of demographic data is not needed. The results shown in Figure 2.2 represent the forecasting accuracy of the crime per capita rate for the 77 community areas present, for the year 2017, using features $F5_1, F4_1, F4_2$.

2.6 Reducing City Crime

The optimal set of features used for crime forecasting includes features that incorporate the per capita crime rates of neighboring communities. We have to ask an important question of which communities have the most influence on the overall crime present in the whole city. To answer this question we propose the deployment of additional crime reduction funding to the communities present. Crime reduction funding can be in the form of increasing home ownership, improving schools, or increasing law enforcement personnel. In [12],[13] we see that an increase in the police force personnel results in a decrease in violent crimes and property crimes. When considering the data on community areas from 2002-2017, we assume that no interventions were made during that period. It is also important to note that we will not try to estimate what the cost of such interventions may be in reality, or whether that cost may vary from one community area to another.

We limit intervention to only one community area at a time, and for that community area

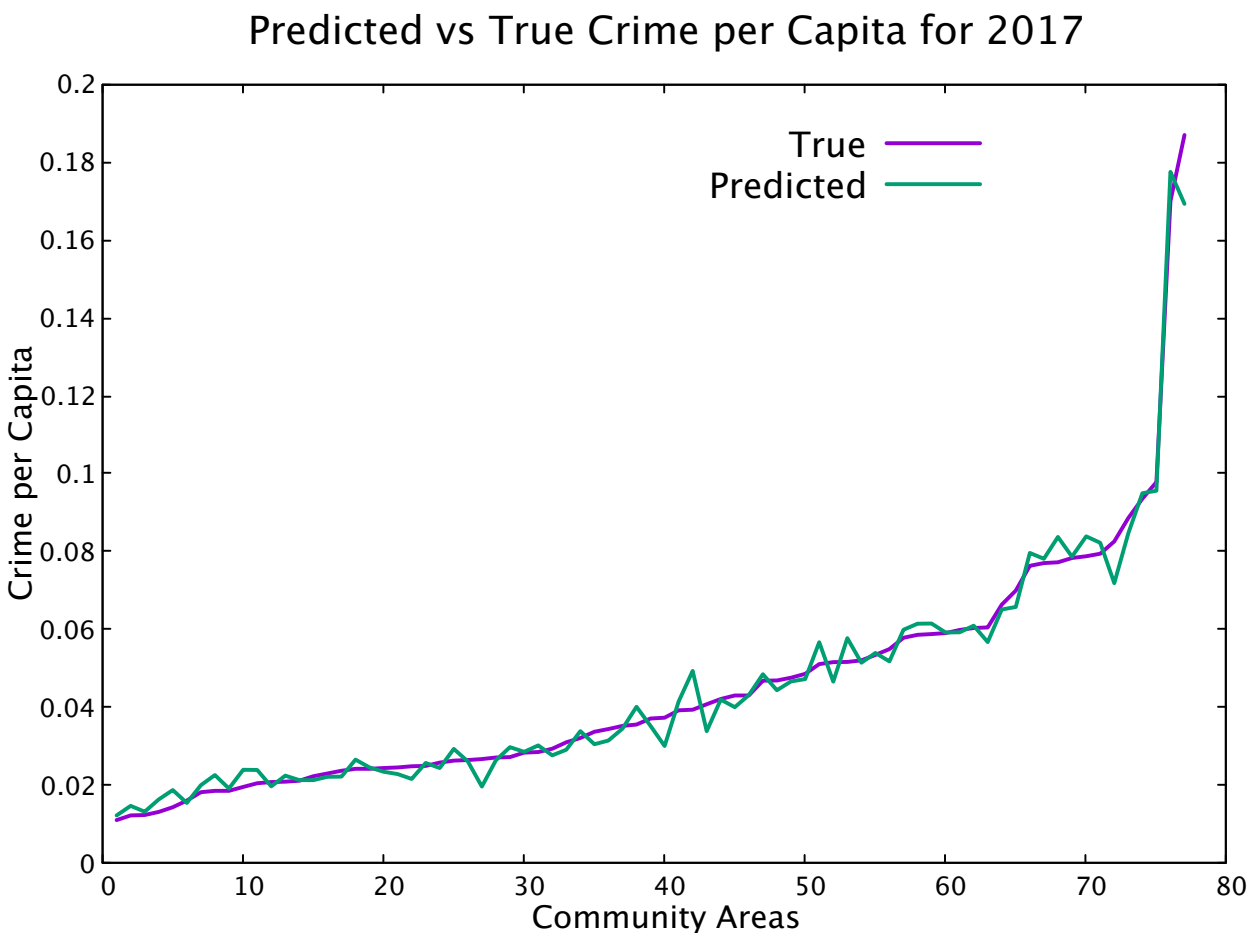

Figure 2.2: Forecasting crime per capita rate for the upcoming year (2017), using features $F5_1$, $F4_1$, $F4_2$. On the x-axis communities are listed in increasing order of crime per capita rate.

the intervention is at a very small scale (a few percent change). This enables us to continue using the models prepared for each year of crime in the city, since we can claim that the models remain valid to formulate predictions, with the proposed crime reduction intervention. We will apply the most simple and straight-forward crime reduction model. The total crime reduction will only be applied to one community, when in reality the crime reduction funding can be spread over several communities. We will also assume that the crime reduction intervention will only be applied for one year, when in reality the intervention can be spread over several years. We apply crime reduction intervention by choosing a community area for deploying intervention, and predicting crime rates for all the community areas for the next year, with the crime reduced in the chosen community.

For our simple one-year one-community intervention, we use the optimal set of features

$F5_1, F4_1, F4_2$, where $F5$ is *CommunityCrimeRate* for the last year previous to the forecast year, and $F4$ is *NeighborhoodCrimeRate* accounting for the neighborhood crime rate for the last two years before the forecast year. Then for each community, we run a simulation, and we reduce the crime rate for that community for the year 2016 as indicated by the size of the intervention.

2.6.1 Methodology

Let $C_c(t)$ denote the total number of crimes in the city for year t. Let $P_c(t)$ denote the total population of the city in the year t, and let $d_n = 77$ stand for the number of community areas present in the city. With that, we have the average crime rate in the city as $C_c(t)/P_c(t)$, and the average population of a community is $P_c(t)/d_n$. Taking their product, $C_c(t)/d_n$, gives us the average number of crimes in a community with an average population size. We define the intervention size in terms of the number of crimes reduced in a community from the above value. Let the rate of reduction be denoted by $r = 0.025$, So we get $V_r(t) = rC_c(t)/d_n$. This helps us guarantee that the intervention size is constant, and does not depend on the crime rate of a given community, or the population size of a given community. With this, we get that the initial number of crimes reduced in a community d is $I_{d,r} = \min(V_r(t)/r, 1)$.

There are three types of communities from the point of view of the intervention. First, the community in which intervention is directly applied. Second, communities that neighbor the community in which intervention was applied. Third, and finally, communities that intervention is neither directly applied to, nor applied to any of their neighboring communities. To compare results between the three types of communities, we have to compute the branching rate of crime change in a given community d. This branching rate for community d is computed as the multiplier metric, denoted by $M_{d,r} = T_r/V_r(t-1)$ for year t, where T_r denotes the overall number of crime incidents in the city for year t under intervention with rate r in community d during year $t-1$.

1. The first case is communities where intervention is directly applied. For such communities the values for features $F4_1$ and $F4_2$ remain unchanged, since there is no change to the neighboring communities. What will change is the value of feature $F5_1$. Let $\Delta C_{i,t+1}$ denote total crimes reduced by intervention in the year $t+1$ for community i. The presence of additional funding during year t, results in the decrease of I crimes in that year in community i. The post-intervention rate of crimes for this community now becomes $(C_{i,t} - I)/P_{i,t}$. Therefore

the crime rate changes as follows.

$$F5_1\left(\frac{C_{i,t}}{P_{i,t}}-\frac{C_{i,t}-I}{P_{i,t}}\right)=\frac{F5_1 I}{P_{i,t}}. \tag{2.3}$$

Hence, the actual change in the number of crimes is

$$\Delta C_{i,t+1}=F5_1 I\frac{P_{i,t+1}}{P_{i,t}}. \tag{2.4}$$

2. For a community n neighboring community i, the value for features $F5_1$ and $F4_2$ remain unchanged. The only feature whose value will change is feature $F4_1$ in the year t.

$$\Delta C_{n,t+1}=F4_1 P_{n,t+1}\left(\frac{\sum_{k\in N_n}C(k,t)}{\sum_{k\in N_n}P(k,t)}-\frac{\sum_{k\in N_n}C(k,t)-I}{\sum_{k\in N_n}P(k,t)}\right)=F4_1 I\frac{P_{n,t+1}}{\sum_{k\in N_n}P(k,t)}, \tag{2.5}$$

where N_n denotes neighbors of community n.

3. Finally, for all the remaining communities that intervention is neither directly applied to, nor applied to any of their neighboring communities, nothing changes in values of features for years $t-1,t$. Since there is neither a change for such a community nor for its neighbors, there is no change of prediction for this community for year $t+1$.

Summing up the changes from cases 1 and 2, we get formula for $M_{i,t+1}$ as

$$M_{i,t+1}=F5_1\frac{P_{i,t+1}}{P_{i,t}}+F4_1\sum_{n\in N_i}\frac{P_{n,t+1}}{\sum k\in N_n P_{k,t}} \tag{2.6}$$

2.6.2 Results

We start conducting our experiments, by running the above crime reduction intervention on each of the 77 communities. We reduce crime for a given community for the year 2016, and observe how that change affects the overall city crime for the year 2017. One by one we applied intervention to each community, while leaving all the other communities unchanged, to see which of our communities had the highest branching rates, and which of our communities had the lowest branching rates. Table 2.3 shows the three most influential community areas, and Table 2.4 shows the three least influential community areas. We see that between the most influential community, and the least influential community, the branching rate of crime change drops by 50%. It is important to note, that all the communities, regardless of how influential they are, reduce the overall city

crime, when crime reduction funding is applied to them.

Table 2.3: The three most influential communities in decreasing order.

Community ID	$V_r(t)$	$M_{d,r}$	Number of crimes reduced in community with intervention	Number of crimes reduced in the city
62	37.64	2.737	42.22	103.03
4	37.64	2.685	42.88	101.07
29	37.64	2.542	42.05	95.70

Table 2.4: The three least influential communities in decreasing order.

Community ID	$V_r(t)$	$M_{d,r}$	Number of crimes reduced in community with intervention	City wide crime reduction in 2017
30	37.64	1.554	42.69	58.51
56	37.64	1.439	42.82	54.19
74	37.64	1.378	42.52	51.86

2.6.3 Conclusion and Limitations

From the results presented in Tables 2.3 and 2.4, one can see that some communities can in fact be much more influential, in terms of reducing overall crime, in comparison to others. The difference in the branching rate between our most and least influential community is about 0.5. These results emphasize that any form of intervention must be carefully planned, to attain the best possible results.

One glaring limitation of our work is that city wide crime reductions are based solely on predictions, with no real way of testing the predictions in reality. Which is why we applied very small interventions, to maintain the validity of our prediction algorithms. The actual branching rate values are not what is important in our study. The important aspect of this study was finding if some community areas are more influential than others. Therefore, even if the branching rates were to change, the presence of more and less influential communities should persist.

CHAPTER 3
GENERATING AND SHUFFLING SOCIAL NETWORKS

The contents of this chapter are based on the work published in [14] and [15], and contains the parts of the publication that were originally contributed by the author.

3.1 Social Search in Shuffled Social Networks

3.1.1 Overview

Social networks are abundantly available in today's world, and can help connect users from across the globe. The idea of a small world, which was proposed by Milgram in the 1960s, depicts how people can utilize their direct connections to efficiently perform a global search in surprisingly few steps. To simply exemplify how a social search may occur, one can imagine a person looking through levels of his/her friendships to try and connect with another person that they have no direct connection to. We use a real location-based social network, Gowalla, to perform a similar synthetic social search. The results observed in this chapter help us conclude that the physical location of a user is of no importance to the social search efficiency. In fact, what is of actual importance is the spatial distributions of friends. We also show that some knowledge of i-friends (we will refer to them as indirect friends or *i-friends* in short) significantly improves the social search efficiency.

3.1.2 Introduction

Social search as a problem, involves tasking a source user, to try and reach a target user, using their direct social connections. One of the more popular experiments of social search is the Milgram's small world experiments [16]–[18]. We applied the same idea used behind Milgram's experiment to the Gowalla social network. We used the Gowalla network to perform a synthetic social search experiment, see Fig. 3.1. Gowalla contains exact geographical locations of users, by allowing its users to 'check-in' at different locations, and share their geographical coordinates with their friends on Gowalla.

The majority of social networking sites today have an average of six degrees of separations

Portions of this chapter previously appeared as: A. Elsisy, B. K. Szymanski, J. A. Plum, M. Qi, and A. Pentland, "A partial knowledge of friends of friends speeds social search," PLoS One, vol. 16, no. 8, Aug. 2021, art. no. e0255982.

Portions of this chapter previously appeared as: A. Elsisy, A. Mandviwalla, B. K. Szymanski, and T. Sharkey, "A network generator for covert network structures," Inf. Sci., vol. 584, no. 1, pp. 387–398, Nov. 2021.

between most of their users, (4.7 in Facebook [19], 3.7 in Myspace [20], 4.1 in Twitter [21], and so on [22]). The social network that we used, Gowalla, is a location-based social network, which allowed us to track the locations of users, and their friends. This enhanced our understanding in regards to how users in real social networks maintain links, and the properties of friendships created and maintained on modern day social networks.

The availability of geographic coordinates for each user in the Gowalla network allows for the analysis of the spatial distributions of friendships between users. It was found that in the Gowalla social network about 35% of the friends of an average user are geographically located within a 160 km radius of that user's location [23]. Analysis of the spatial distribution of i-friends in Gowalla revealed that about 20% of i-friends are also on average within a 160 km radius of an average user's location. As for higher order of friendships, such as friends of i-friends, the fraction drops to below 5% for the same radius. The spatial distribution of friends can vary based on areas, as it was found that the spatial distribution of friends significantly differs inside and outside metropolitan areas [24]. It is very likely for a person to know some of the friends of his or her friends, but it is very unlikely that a person will know of the friends of his or her friends. To account for this notion, we present a partial knowledge of i-friends when running social routing experiments, using a partial knowledge parameter κ. κ defines the maximum number of i-friends known to a user for each of his/her friends. We run several experiments and find that having some partial knowledge of i-friends, about 15, drastically improves the social search efficiency.

Finally, we pose two novel questions. The first is how much social search is affected by changes to κ when forwarding decisions are based on partial knowledge of i-friends. We find that increasing κ strongly and positively influences the performance of the social search only if κ is small. The second novel question is how the different ways of distributing friendship edges, and use of partial knowledge of i-friends influence the Milgram social search. Surprisingly, we find that only the distribution of friendship edges affects efficiency of this search. Hence, preserving only this distribution and using partial knowledge of i-friends are necessary and sufficient conditions for efficient social search.

3.1.3 Model

To proceed with our social search experiment, we had to decide on the criteria that a user would use to select the friends which they will choose in the forwarding chain, in order to reach the target user. For a user that has to choose a successor, they will compute a utility score of all

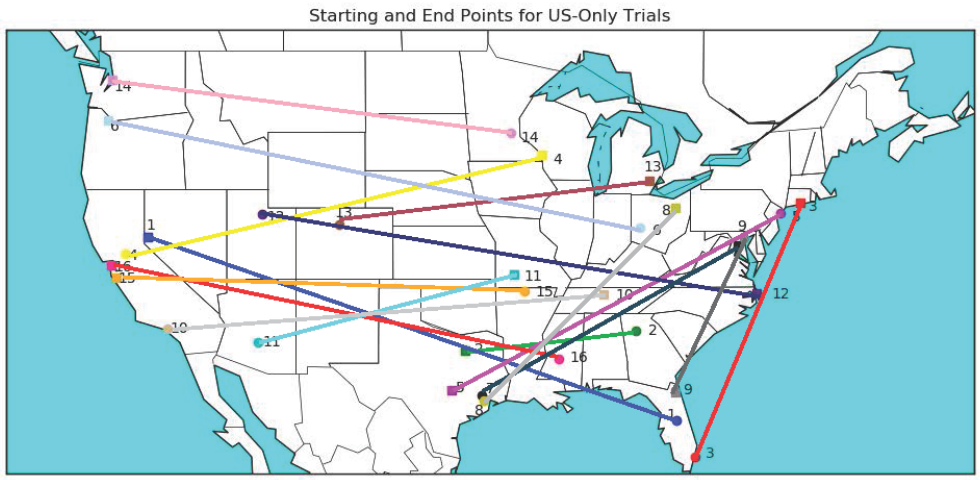

Figure 3.1: Social search experiments in the Gowalla network. Starting nodes are shown as circles, and target nodes are shown as squares. Source and target nodes of the same color represent a pair of nodes which we want to connect through levels of friendships.

their friends, and ultimately choose the friend with the highest utility score. For every friend i, of user U, utility score is computed as:

$$U_i = W_D * D_m + W_C * C_m + W_P * P_m \tag{3.1}$$

where W_D, W_C, and W_P are weights represented by real numbers in the range of $[0,1]$, with the constraint that the total sum of $W_D + W_C + W_P = 1.0$. These weights correspond to the following metrics, distance, community size, and prominence. Normalized distance metric, D_m denotes the normalized distance between the locations of node i and the target, the lower this distance is, the higher the score of friend i. Community metric, C_m, defines the normalized size of the community to which both the node i and the target belong, the smaller the size of the community shared between i and the target node, the higher the score of friend i. Prominence metric, P_m, denotes the normalized degree of node i, the more prominent i is, the higher its prominence score. We tried several configurations using the mentioned three metrics. The first configuration, with all weights equal to zero, randomly chooses a friend to forward the folder to, and it serves as the baseline. The next three configurations each have a value of 1.0 for one of the metrics, and a value of 0.0 for the remaining two metrics. The fifth, and optimal, configuration has weights

$W_D = 2/24, W_C = 7/24, W_P = 15/24$. We find the optimal set of weights by conducting a binary search starting with $W_D = 1/3, W_C = 1/3, W_P = 1/3$, and then varying each weight by a step size of $1/6$, and we proceed be decreasing the step size by half at each step, while maintaining that the sum of all weights is equal to 1.0. With the optimal set of weights found, we now had to find the optimal value for the partial knowledge parameter κ. We used the aforementioned five configurations to measure the impact of κ on the success rates achieved. Starting with $\kappa = 0$, and slightly increasing it for each run, we find that as κ keeps increasing its impact on the success rate diminishes, and found the optimal κ to be equal to 15. With the optimal weights, and $\kappa = 15$ we achieved a success rate of 94%, so higher than the 77.8% rate achieved without it. The difference is significant from the perspective of the failure rate, which is 6% in the first case but 22.2%, so over three times higher than in the first case, see Figure 3.2.

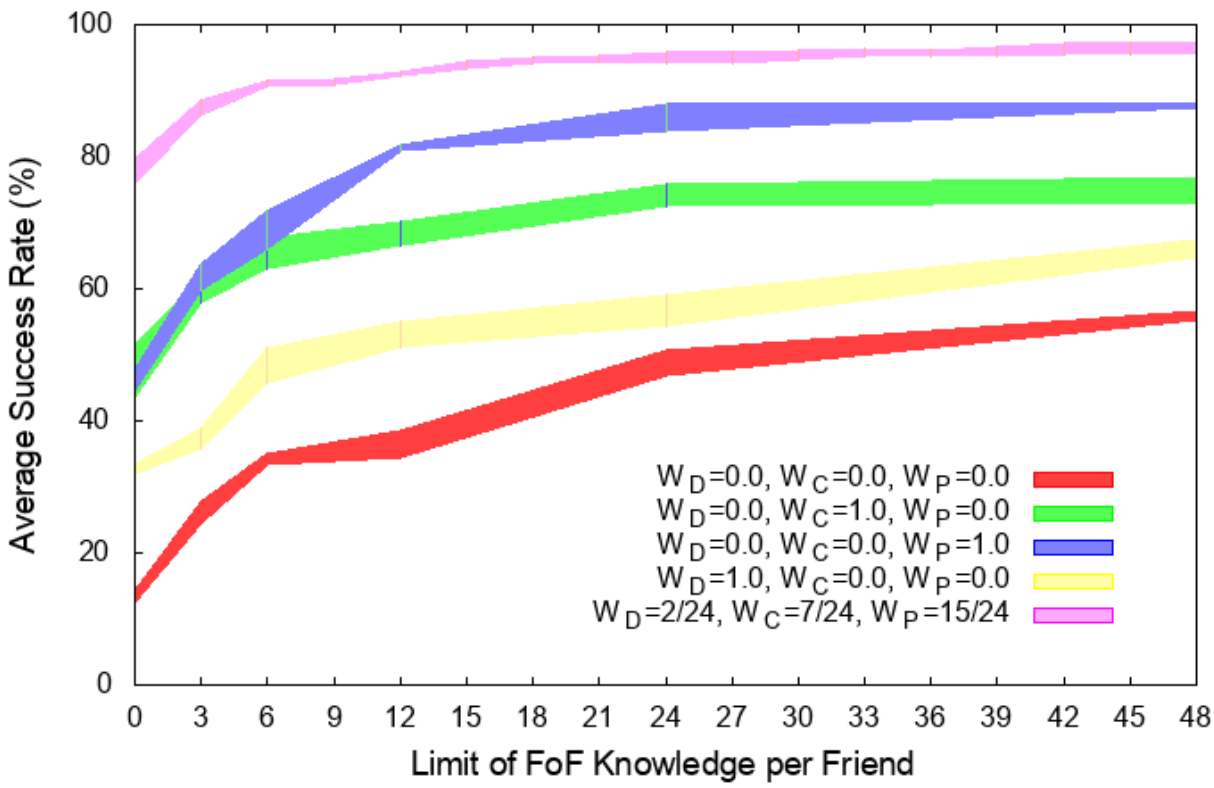

Figure 3.2: We show the success rates achieved using different combination of weights for W_D, W_C, and W_P, and show that the optimal set of weights are $W_D = \frac{2}{24}$, $W_C = \frac{7}{24}$, and $W_P = \frac{15}{24}$. Success rates are shown as a function of the maximum number of i-friends allowed, with error bars showing the standard error.

It is also important to note that in all our experiments we only covered the contiguous US territory (i.e. excluding Alaska and Hawaii). For the experiments to follow, we had to divide the

contiguous US territory into equal sized rhomboids, with sides of approximately 70 km. The sides of the rhomboids were drawn along meridians and parallels, thus simplifying the translation of geographical coordinates into positions in the rhomboids and vice versa. To cover the contiguous US territory, we had to create 1,860 rhomboids, but many of which were empty with no Gowalla users residing in them. The empty rhomboids were removed, many of which were in lakes or oceans, and we were left with 850 rhomboids.

More on the process of the removal of empty rhomboids. Note that for each rhomboid, we have the following: on-boundary flag, and a vector of size four set to $external = (0,0,0,0)$, with each element referring to a boundary (West, South, East, North). In addition to this, we also know how many rhomboids existed in each row and in each column. Please refer to Algorithm 1 for the full pseudo code.

3.1.4 Experiments

To answer the questions we set out to answer, it is necessary to do the following. To understand the impact of the spatial embedding of nodes into space on social routing efficiency, we use eight different methods of embedding nodes into space. To understand the influence of friendship distributions among nodes on social search efficiency, we use five different methods of assigning node degrees to our users.

3.1.4.1 *Embedding of Nodes*

The eight node distributions used are the following. First, original node distributions, to be used as the baseline to compare against. Second, random node distribution, which assigns random locations to each Gowalla user, by randomly choosing a rhomboid, and randomly picking latitude and longitude coordinates within that rhomboid to place the user. The final six distributions are a combination of one of three methods of distribution nodes across rhomboids, and one of two methods for embedding the node into actual coordinates within the rhomboid. The first three distributions for distributing nodes across rhomboids are, exponential distribution, normal distribution, and Zipf distribution. All three distributions use the mean of the populations of the rhomboids in the original data. Additionally, the normal distribution uses the variance across the rhomboids population in the original data. Finally, the Zipf distribution starts with the value of 10,700, which is the value of the largest population across all the rhomboids in the original data, which generates a total population that is closest to the total population of the original Gowalla

Algorithm 1 Rhomboids Removal Algorithm

```
Step 1: Initialization
  for each rhomboid r in left-most column do
      on-boundary[r] = 1
      external[r][0] = 1
      if rhomboid r population == 0 then
          add rhomboid r to unprocessed queue of rhomboids
      end if
  end for
  for each rhomboid r in right-most column do
      on-boundary[r] = 1
      external[r][2] = 1
      if rhomboid r population == 0 then
          add rhomboid r to unprocessed queue of rhomboids
      end if
  end for
  for each rhomboid r in lower-most column do
      on-boundary[r] = 1
      external[r][1] = 1
      if rhomboid r population == 0 then
          add rhomboid r to unprocessed queue of rhomboids
      end if
  end for
  for each rhomboid r in upper-most column do
      on-boundary[r] = 1
      external[r][3] = 1
      if rhomboid r population == 0 then
          add rhomboid r to unprocessed queue of rhomboids
      end if
  end for
Step 2: Removal
  for rhomboid r in unprocessed queue do
      for each element in external vector of rhomboid r do
          if external[i] == 0 then
              select rhomboid rn (neighboring rhomboid in the direction defined by i)
              if population of rn == 0 and on-boundary of rn == 1 then
                  push rn on the unprocessed queue
              else
                  if population of rn != 0 and on-boundary of rn == 1 then
                      external[i+2%4] = 1
                  end if
              end if
          end if
      end for
  end for
```

network. The two methods for embedding nodes into rhomboids are geographic, and uniformly random. The geographic method looks at the original rhomboids, and finds the rhomboid with the closest population as the rhomboid that the user is about to be placed in, and places the user using positions occupied by real users in the original rhomboid that was chosen. The second method, uniformly random, places users uniformly randomly into the rhomboid. For each of the mentioned distributions, except original, we generated 10 samples. We ran each created sample 10 times and averaged the results.

3.1.4.2 *Distribution of Friends*

The five methods for assigning node degrees to Gowalla users are the following. First, original friendships, which will be used as the baseline for comparison. Second, random distribution while preserving both the node degree and the distance range of a node's friend, while randomizing the actual friends. To preserve the node's degree and the range of friendships for each node, each node swapped edges with its neighbors that are in the same rhomboid with friends in the same range of distances to those nodes. The third method, random uniform, generates a friendship distribution using Erdos-Renyi [25] random graph, with an average node degree as the original Gowalla network. The fourth method, exponential distribution, with a mean degree the same as the original Gowalla network. Our fifth and final method, Power Law distribution, with a mean degree the same as the original network, and a range of node degrees specified so the total number of nodes very closely matches the total number of nodes in the original network. We again created 10 samples for each of the mentioned distributions, except for the original distribution. We ran each created sample 10 times and averaged the results. Then, we implemented and ran the elch's [26], two-tailed, t-test to check whether or not the differences in performance are statistically significant, and we report the results below.

More on the random friendship distribution while preserving the node degree. First we will look deeper into how we preserve each node's degree. Let us assume that we have two source nodes S_1 and S_2, and the only condition we have for these two nodes is that they cannot be friends in the original network. We now select two target nodes, $T_!$ and T_2, where T_1 is a friend of S_1, and T_2 is a friend of S_2, not only that, but also very importantly that T_1 and S_2 are not friends, and that T_2 and S_1 are also not friends in the original network. We now simply shuffle the edge. Originally we had (S_1, T_1), and (S_2, T_2), and after shuffling we have (S_1, T_2), and (S_2, T_1). The final condition we have is that any two nodes can only share one edge, so an edge cannot be assigned more than

once between the same two nodes.

We now focus on the distance preservation part of our proposed friendship distribution. For each of our users, we computed the distribution of their friends through each neighborhood, with a total of nine neighborhoods present in our network. The first neighborhood is the set of rhomboids that share an immediate border with the rhomboid that our user resides in. The second neighborhood is the set of rhomboids that share borders with the rhomboids in the first neighborhood, and the set of all rhomboids that share borders with those rhomboids. This pattern continues for each neighborhood further out as we increase the step size. Our neighborhood distance steps are d = [1,3,7,15,31,63,127,255,511]. After the completion of this step we should have for each user, the distribution of their friends through each neighborhood. Our second step is to compute the populations of the neighborhoods of each rhomboid, which can be done as follows. First, create a population matrix representing the rhomboids populations in our data. Second, create a left population matrix by going through each row, and each cell in that row will have the population of itself plus the population of all the cells in the same row to its left. Third, create an up-left population matrix, where each cell will now have its population, plus the population of all the cells to its left and up of it. With this set up, the calculation of neighborhood populations is now relatively simple, and extremely efficient. To compute the k-neighborhood population density of each rhomboid, we will need the population of four rhomboids, which are specified in Procedure 1. Once we have gathered the populations of the four required rhomboids, we can proceed to finding the k-th neighborhood population of our given rhomboid, which will be equal to:

$$P_k = R_1 - R_2 - R_3 + R_4 \tag{3.2}$$

, where P_k denotes the k-th neighborhood population of our given rhomboid, and R_1, R_2, R_3, R_4, denote the populations of the four rhomboids specified earlier.

We now know for each user the rhomboids that the user's friends reside in, and we know the users that reside in each rhomboid's neighborhoods. We proceed by doing the following: for each rhomboid, we go through the users residing within this rhomboid. For each of those users, we go through their friendship edges, and we find the neighborhood that each friendship edge goes to. We then look at the nodes residing in the neighborhood of that friendship edge, and if a node within that neighborhood has an edge going to the source rhomboid, then we can shuffle the edges, the same way as explained earlier. This shuffling of edges preserves the degree, and also preserves the range of distance of the friendships.

Procedure 1 Computing Population of Neighborhoods of Rhomboids

R_1: Will have the population of all rhomboids in the k-neighborhood, and possibly other neighborhoods that we do not care about.
row = source rhomboid row + d[k]
column = source rhomboid column + d[k]

R_2: Will have the population of all rhomboids in neighborhoods outside our range and residing to the top of our rhomboid. If row value is less than 0, R_2's population will be set to 0.
row = source rhomboid row - d[k] - 1
column = source rhomboid column + d[k]

R_3: Will have the population of all rhomboids in neighborhoods outside our range and residing to the left of our rhomboid. If column value is less than 0, R_3's population will be set to 0.
row = source rhomboid row + d[k]
column = source rhomboid column - d[k] - 1

R_4: If row or column values are less than 0, R_4's population will be set to 0. Also if R_2 or R_3 had population set to 0, then R_4's population will also be set to 0.
row = source rhomboid row - d[k] - 1
column = source rhomboid column - d[k] - 1

3.1.5 Success Rate and Stretch Using Partial Knowledge of Indirect Friends

As aforementioned, it is safe to assume that a person has some knowledge of the i-friends, but it is highly unlikely that a person would know everything about their i-friends, or that they would know absolutely nothing about their i-friends. Which leads us to the question of how much can we expect an average person, or user in our Gowalla network, to know about their i-friends. An average user in our Gowalla network has 12 friends, but surprisingly has 2,016 unique i-friends. To limit the number of i-friends that our average Gowalla user knows, we limit each user to 15 i-friends, which are randomly chosen, for each of their friends. This brings down the average number of i-friends from 2,016 to 125.3, which is a lot more reasonable.

For the three metrics we used: distance, prominence, and community, it is reasonable to assume that a user would have some partial knowledge on these metrics for the friends of his/her friends. Regarding the distance metric, we can expect a user to have some partial knowledge on past locations that their friends have lived at, or travel destinations that they recently or frequently visited. For communities, it is reasonable to assume that a person would know some attributes of their friends, for example: hobbies, profession, and thus can have some partial knowledge on which communities their friends are involved in. As for the prominence metric, it is logical to

assume that a friend would mention if they were friends with a famous or notable person, which a person would recognize and/or remember. These are all examples of partial knowledge of i-friends that can allow a user to choose their successor friends based on some knowledge that they have of the friends of that friend.

Table 3.1: Distributions of fractions of friends, i-friends and communities over the ranges of distances from nodes. The second column shows the fractions of friends at each distance range, computed by summing the numbers of friends in each range for each individual user and then dividing the result by the total number of friends of all users. The third column shows fractions computed as ratios of sums of the numbers of i-friends of each user at each distance range to the total number of i-friends for all users. The fourth column shows fractions of members of communities of each individual user at each distance range listed, computed as the sum of these numbers divided by the total number of members of all relevant communities. Fifth, sixth and seventh columns list cumulative values from the second, third and fourth column, respectively.

Range	Percentage of			Cumulative Percentage of		
(km)	Friends	i-friends	Commu-nites	Friends	i-friends	Commu-nities
≤ 6.25	18.6	2.6	14.0	**18.6**	2.6	14.0
6.25 – 12.5	8.6	1.3	4.3	**27.2**	3.9	18.3
12.5 – 25	10.3	1.7	5.5	**37.6**	5.6	23.9
25 – 50	7.6	1.5	4.6	**45.2**	7.1	28.5
50 – 100	3.9	1.0	2.6	**49.0**	8.1	31.1
100 – 200	3.8	1.6	3.1	**52.8**	9.7	34.1
200 – 400	6.4	6.0	6.8	59.2	15.7	**40.9**
400 – 800	6.4	8.8	8.2	65.6	24.5	**49.1**
800 – 1600	11.8	23.8	17.2	77.4	48.3	**66.4**
1600 – 3200	14.8	36.4	23.3	92.2	**84.7**	**89.7**
3200 – 6400	7.5	14.2	10.0	99.8	**98.9**	99.7

Let us dive more into understanding how partial knowledge of i-friends impacts the three metrics we are using. Starting with the distance metric, we observed that the average distance between our starting and target node is 1,880 km. We find that 49% of direct friends of an average user, are within a 100 km radius of the user's location. Meanwhile, for an average user, only 8.1% of their i-friends are within a 100 km radius, while 59% of their i-friends are within 800 to 3,200 km from the user's location, see Table 3.1. With an average distance of 1,880 km between our starting node and target node, and 36.4% of a user's i-friends being in the radius of 1,600 to 3,200 km away from the user. With an average of 125.3 i-friends, and 36.4% of i-friends being within

that range, a user has about 45.6 i-friends within this range with the possibility of the distance of the closest i-friends to the target user being 280 km. Using only direct friends, it would take on average 3 or more steps to get this close to the target user.

For the community metric, we compute the following. The average number of communities that a user belongs to, which is 1.0, the average number of communities that a user can have access to through friends, which is 6.8, and the average number of communities that a user can reach through i-friends, which is 587. With κ limited to 15, a user can still reach about 47.4 communities through their i-friends, and when we account for the two steps required to reach those communities of i-friends, the user would still have access to about 3.5x more communities, than they would access through two steps using two of their friends.

For the prominence metric, and the usage of i-friends, we account for the following. First of all, we separate our users into prominent, and non-prominent, with prominent users being within the top 1% of all nodes in terms of prominence (degree). We have 758 prominent nodes in the Gowalla network, each with a degree higher than 122. Such nodes are unlikely to look at their i-friends when trying to find a popular user, as they themselves already have a lot of friends. Non-prominent nodes have on average 9.2 friends, 2.1 of which are prominent, and 2,016 i-friends, 188 of which are prominent. With $\kappa = 15$, non-prominent nodes have about 75 i-friends, 15.4 of which are prominent. Accounting for the two steps needed to reach that i-friends, a user has a reach that is 3.5x stronger to prominent nodes, than they would through two steps using their direct friends only.

Looking at the results of the social search simulations run on the original Gowalla network, we make some very interesting observations. First of all, note that the optimal weights used are $W_D = 2/24, W_C = 7/24, W_P = 15/24$, with a ratio of $1 : 3.5 : 7.5$, so the prominence metric is more than twice as important as the community metric and more than seven times as important as the distance metric. We find that for the social search experiments conducted on the original Gowalla network, 60% of the hops used direct friends, and 40% used i-friends. It is also important to note that even though we allow for up to 15 i-friends for each friend, an average of only 3.51 i-friends were used, see Table 3.2. We were also interested to know which of the three metrics were most dominant when selecting the recipient node. A metric dominating is simply stating that this metric formed the largest component of the total score of the selected node. We find that for the hops using direct friends, 0.3% of the hops were dominated by the distance metric, 22.7% by the community metric, and 77.0% by the prominence metric. Meanwhile, for the hops using i-friends, 0.6% of

the hops were dominated by the distance metric, 30.1% by the community metric, and 69.3% by the prominence metric. Of the 40% of the hops involving i-friends, when the package is sent to a mutual friend to forward it to the i-friends, who is a direct friend of the mutual friend, 74% of the time the mutual friend sends the package to the intended i-friends. Meanwhile, 4% of the time the mutual friend finds a better direct friend, who is unknown to the original sender, and sends the package to them. And 22% of the time the mutual friend will send the package to an i-friend of their own.

Table 3.2: The numerical values of the average number of i-friends used, with different distributions of nodes and friendships. We limit the knowledge of i-friends to 15 per friend. We find that for the original friendship distribution (original Gowalla friendships), regardless of which spatial distribution of nodes is used, the average i-friends used are around four. Average i-friends used is computed as the number of hops using i-friends, divided by the total number of hops needed to get from the source user to the target user. All reported results are for $W_D = \frac{2}{24}$, $W_C = \frac{7}{24}$, and $W_P = \frac{15}{24}$, with $\kappa = 15$.

Friendship distributions: Node distributions	Original	Random	Exponential geographic	Exponential uniform	Normal geographic	Normal uniform	Zipf geographic	Zipf uniform
Original	3.51	3.76	3.94	3.78	3.94	3.77	3.64	3.83
Random degree-preserving	9.27	17.03	14.45	15.49	15.81	13.06	13.17	10.32
Random uniform	24.73	24.74	24.38	24.03	24.54	24.48	24.51	23.57
Random exponential	24.38	23.31	24.02	23.35	24.48	23.87	24.32	23.30
Random Power Law	25.80	26.09	26.53	27.03	26.30	26.66	26.44	26.67

As mentioned, we find that on the original Gowalla network, with 15 i-friends limit per friend, an average of 3.51 i-friends are used. Meaning, that on an average path from a source node to a target node, only a few number of i-friends are used when deciding which user to choose as the recipient. About 60% of the hops used direct friends, thus showing that the direct friends often outscore the i-friends. When an i-friend is chosen as the next recipient, even though a user does not have full knowledge about all the friends of their friends, 74% of the time, the package reaches the intended i-friends, through a mutual friend. This shows that a chosen i-friend is more often than not the best option out of all the direct friends of the current user, and all of the friends of the friends of the current user. Thus confirming two things, firstly, the usage of i-friends allows us to make the social search much more efficient, as the chosen i-friends usually have a much higher

score than any direct friends. Secondly, increasing the limit of i-friends knowledge to more than 15 does not help improve the success rate by much, as shown in Figure 3.2, as the intended i-friends outscores all the other possible recipients 74% of the time.

The original set of friendships achieved the highest success rate that, surprisingly, is independent of the way nodes are distributed over the space. The success rate with partial awareness of i-friends for $\kappa = 15$ is statistically significantly higher than the success rate with $\kappa = 0$ (no knowledge of i-friends) and all other friendship edge distributions (P-value$= 0.0005$). The distant second is the random friendship edge distribution preserving the degree and the ranges of distances from friends of each node, whose success rate is not statistically significantly higher for this distribution combined with $\kappa = 0$ (P-value$= 0.2481$), but it is statistically significantly higher for the remaining tested methods of friendship edges distributions (P-value≤ 0.0005). The remaining three methods of distributing friendship edges achieve much lower success rates. Accordingly, their success rates without knowledge of i-friends are even lower and since they drop below 10%, they are not shown.

Now focusing on the differences between hops using direct friends, and hops using i-friends. In both cases, the distance metric is of very low importance in comparison to the community and prominence metrics. Nonetheless, it is still important to note that the distance metric was twice as dominant for hops using i-friends as it was for hops using direct friends. This shows that, as we hypothesized, the usage of i-friends can be of significance when trying to reach a target user that is distant from the current user, as i-friends are often geographically distant, as shown in Table 3.1. We also find that the community metric is dominant for 30.1% of the paths using i-friends, and for 22.7% of the paths using i-friends. Showing that through the usage of i-friends, we have access to more diverse communities, which increases the chances of finding a community that the target user is present in. Even though for the original Gowalla network i-friends are not frequently used, their usage is deemed beneficial as they provide a "shortcut" to reaching the target user. This is due to their presence in different communities than the communities of the direct friends, as many of the direct friends will belong to the same set of communities. For our third and final metric, we find that prominence is dominant in 77% of the hops involving direct friends, and in 69.3% of the hops involving i-friends. This is the only metric that is more dominant for hops using direct friends. The usage of i-friends is of most importance when trying to reduce the distance to the target, or when trying to diversify the communities that we are searching through, but not as important when trying to find a recipient that has high prominence. Even though i-friends can have higher prominence than direct friends, in most of the cases an i-friend with high prominence would imply that their

direct friends are also of high prominence.

In Fig. 3.4(b) we also show the stretch with and without i-friends awareness for the second best performing distribution of friendship edges that again is the original and the random preserving nodes' degree and ranges of distance to their friends. The stretch of the original friendship edge distribution with $\kappa = 15$ is statistically significantly lower than that of this distribution with $kappa = 0$ (P-value$= 0.0051$), and the remaining friendship distributions (P-value≤ 0.0005). Similarly, the stretch of the second in performance random friendship edge distribution preserving the degree and the ranges of distances from friends of each node is not statistically significantly higher for this distribution combined with $\kappa = 0$ (P-value$= 0.5038$), but it is statistically significantly higher for the remaining tested methods of friendship edges distributions (P-value≤ 0.0053).

In conclusion, when i-friends awareness is used, the distribution of nodes does not affect the stretch, but the friendship edge assignment does.

3.1.6 Methodology

The location-based social network used in this study is Gowalla [27]. At the time of the collection of this network, Gowalla was mainly used in the United States, and in Sweden, with a total of 154,557 users, and 1,139,110 friendships. To ensure connectivity in our network, we only considered users within the United States contiguous territory, which consisted of 75,803 users, and 454,350 friendship edges, with the node Power Law degree exponent $\gamma \approx 1.49$. In Fig. 3.3, we plot the degree distribution for the giant component of a network comprised of Gowalla users located in the U.S.

For each of the social routing experiments that we conducted, we randomly uniformly selected 100 starting users, and 100 target users from the network, and executed a social search, with a limit of up to 50 hops, with the condition that the starting and target users are at least 1,600 km apart, and the source node having a degree of at least 2. For each hop, the current user chooses one of his/her friends as the successor, in order to ultimately reach the target user, with the condition that this successor has not been chosen previously, to avoid any loops. The search can either fail, or can successfully end once the target user has been reached.

Regarding the three metrics discussed earlier, here we will formally define how the value of each metric is computed. For the distance metric, let D_{max} denote the diameter of the network, and D_i denote the distance between node i and the target node. If there is a community to which both the target and the current holder of the folder belong, we set C_{max} to the size of this community,

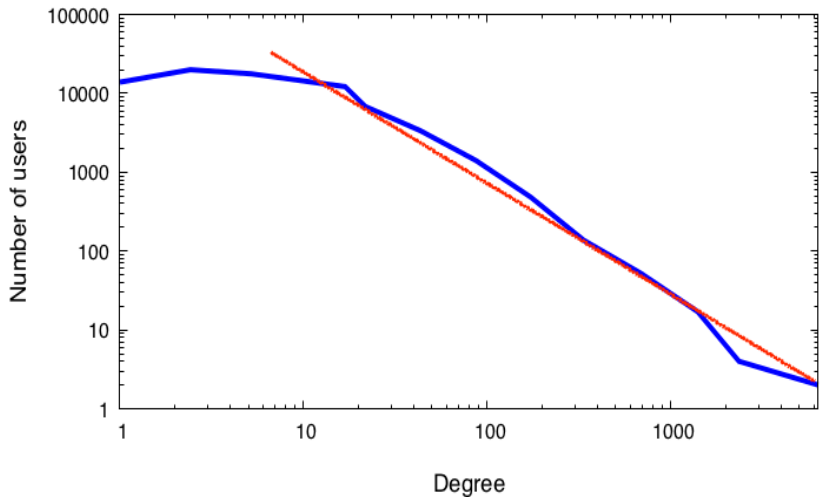

Figure 3.3: The degree distribution for the giant component of a network with 75,803 Gowalla users located in the U.S., 454,350 friendship edges, and $\gamma \approx 1.49$.

otherwise C_{max} is set to N, the size of Gowalla network. In our study, Gowalla communities were detected using a label propagation algorithm named Label rank [28]. For the final metric, prominence, let P_{max} be the largest node degree in the network, and let P_i be the degree of node i. The metric values for node i in Eq. 3.1 are defined as follows:

$$D_m = 1 - \frac{\log(D_i)}{\log(D_{max})} \tag{3.3}$$

$$C_m = 1 - \frac{\log(C_{max})}{\log(N)} \tag{3.4}$$

$$P_m = \frac{1}{\log(P_{max}) - \log(P_i) + 1} \tag{3.5}$$

3.1.7 Results and Discussion

The distribution of nodes into space did not have an impact on the success rate of the social search, see Figure 3.4(a), and Tables 3.3 and 3.4. The spatial distribution of friends is what greatly affected the success rate of the social routing experiments. Our highest success rate was achieved using the original friendships, followed by the random degree/range preserving friendships, while the remaining methods all had much worse performance. This shows us that the spatial distribution of friends is what matters, even if the actual friendships are randomized.

We also look at the stretch, which is defined as $S_{n_1,n_2} =< d(n_1,n_2) > /d_s(n_1,n_2)$, where

$< d >$ is the average distance traveled between $n1$ and $n2$ and d_s is the length of the shortest path between $n1$ and $n2$. The results can be seen in Figure 3.4(b), and Tables 3.5 and 3.6. The closer the stretch value is to 1, the better and more efficient the routing is. We observe that the best stretch results are achieved when using the original friendships of Gowalla, followed by the degree and range preserving random friendships. We again see that for the remaining friendship distributions, stretch increases drastically. Unlike before, the spatial distribution of nodes does have some impact on the stretch. From the inset in Fig. 3.4(b), we see that the distribution of nodes has a major impact on stretch when we have $\kappa = 0$, but not that big of an impact when $\kappa = 15$.

Table 3.3: The numerical values of the success rates reported in Fig. 3.4(a), with different distributions of nodes and friendships. All reported results are for $W_D = \frac{2}{24}$, $W_C = \frac{7}{24}$, and $W_P = \frac{15}{24}$, with $\kappa = 15$.

Friendship distributions: Node distributions	Original	Random	Exponential geographic	Exponential uniform	Normal geographic	Normal uniform	Zipf geographic	Zipf uniform
Original	94%	93%	91%	91.4%	92.4%	91.8%	92.8%	91.8%
Random degree-preserving	69.8%	33.8%	45.2%	38.8%	41.6%	58.6%	55.6%	67.2%
Random uniform	26%	30%	30.8%	31.8%	30%	29.8%	30%	32.8%
Random exponential	16.8%	20.4%	18.4%	21.2%	16.4%	21.2%	19%	21.6%
Random Power Law	17.2%	17.6%	13.8%	11.8%	15.5%	13.6%	14.4%	13.2%

Table 3.4: The numerical values of the success rates reported in inset of Fig. 3.4(a), with different distributions of nodes and friendships. All reported results are for $W_D = \frac{2}{24}$, $W_C = \frac{7}{24}$, and $W_P = \frac{15}{24}$, with $\kappa = 0$.

Friendship distributions: Node distributions	Original	Random	Exponential geographic	Exponential uniform	Normal geographic	Normal uniform	Zipf geographic	Zipf uniform
Original	77.8%	72%	70.6%	72%	71.8%	72.6%	73.8%	71.2%
Random degree-preserving	66.6%	30.5%	41.1%	34.9%	40.6%	53%	51.2%	62.6%
Random uniform	3%	4.4%	1.8%	3%	3.4%	4%	2.8%	4.4%
Random exponential	2.4%	2.6%	3.4%	8%	3.6%	2.6%	2.8%	2%
Random Power Law	1.2%	2.4%	1%	1.4%	2.2%	2%	2%	1.4%

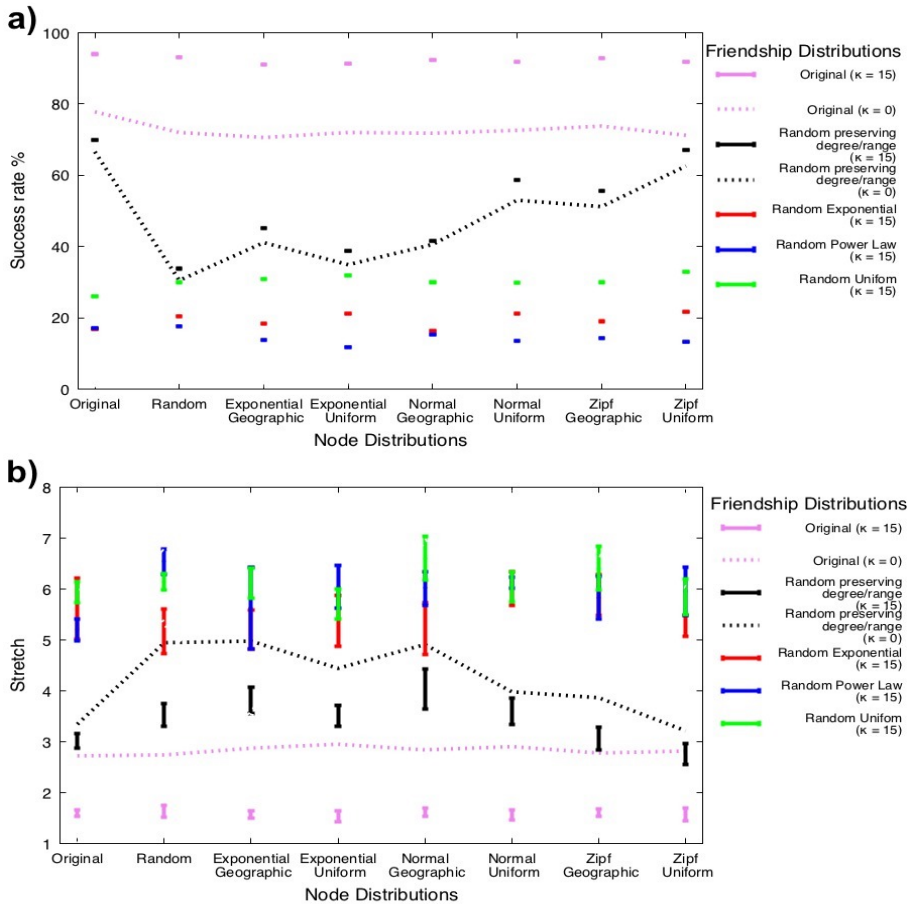

Figure 3.4: (a-b) Show plots with error bars of success rates (a) and stretches (b) achieved with partial knowledge of i-friends under the different distributions of friendship edges as a function of various distributions of nodes into space. Plots represent the results of running 10 samples of each distribution resulting in 100 samples for each case of the considered friendship edge distributions. Each of these samples was executed 10 times and averaged results plotted. Each friendship edge distribution has a unique color assigned to its plots and two best performing distributions have also plots of their stretches achieved without knowledge of i-friends marked with the dashed line. We describe all distributions of friendship to edges and nodes into space in the text. Plots for runs with awareness of i-friends were computed using $kappa$ limit of the number of i-friends set to 15, which, if needed, are uniformly randomly chosen from i-friends for each friend of the sender. The error bars were in the range of [0.002, 0.039] for success rates (a) and in the range of [0.07, 1.61] for stretches (b).

3.2 Synthetic Network Generator

3.2.1 Overview

In the earlier sections of this chapter we focused on social search in social networks, which is based on the work presented in [14]. In the following sections of this chapter we will focus

Table 3.5: The numerical values of the stretch values reported in Fig. 3.4(b), with different distributions of nodes and friendships. All reported results are for $W_D = \frac{2}{24}$, $W_C = \frac{7}{24}$, and $W_P = \frac{15}{24}$, with $\kappa = 15$.

Friendship distributions: Node distributions	Original	Random	Exponential geographic	Exponential uniform	Normal geographic	Normal uniform	Zipf geographic	Zipf uniform
Original	1.82	1.88	1.79	1.75	1.84	1.78	1.83	1.79
Random degree-preserving	3.35	3.92	4.23	3.90	4.47	3.99	3.40	3.06
Random uniform	6.66	6.90	6.86	6.40	7.42	6.78	7.19	6.56
Random exponential	6.34	5.83	5.87	6.06	5.89	6.77	6.64	5.95
Random Power Law	5.75	7.23	6.23	6.68	6.66	6.78	6.44	6.59

Table 3.6: The numerical values of the stretch values reported in inset of Fig. 3.4(b), with different distributions of nodes and friendships. All reported results are for $W_D = \frac{2}{24}$, $W_C = \frac{7}{24}$, and $W_P = \frac{15}{24}$, with $\kappa = 0$.

Friendship distributions: Node distributions	Original	Random	Exponential geographic	Exponential uniform	Normal geographic	Normal uniform	Zipf geographic	Zipf uniform
Original	2.72	2.74	2.87	2.95	2.84	2.90	2.77	2.81
Random degree-preserving	3.34	4.94	4.97	4.44	4.91	3.98	3.86	3.21
Random uniform	7.89	6.72	7.71	4.63	6.94	7.97	6.67	6.04
Random exponential	5.99	6.82	5.60	8.35	7.69	6.22	6.77	4.68
Random Power Law	8.04	5.31	3.48	8.34	6.58	5.24	6.25	6.70

more on the work presented in [15], which focuses on the structures of covert networks, such as terrorist or criminal networks, and the process of generating synthetic networks that are statistically similar to real social/covert networks. Covert networks differ from regular social networks in many ways. The mere fact that the activities of covert networks are illegal, impacts their structures, and the nature of the memberships that they have within. The data collected on covert networks can often be incomplete and/or inaccurate, which results in a challenging process of trying to interpret and understand the structures and activities of such networks. Real data on covert networks is often inaccessible to researchers, due to legal reasons, thus we propose here a synthetic network generator to help address this issue of lack of data. We introduce a novel method of rewiring covert networks parameterized by the edge connectivity standard deviation. The generated networks are

statistically similar to themselves and to the original network. The higher-level organizational structures are modeled as a multi-layer network while the lowest level uses the Stochastic Block Model. Such synthetic networks provide alternative structures for data about the original network. Using them, analysts can find structures that are frequent, therefore stable under perturbations. Another application is to anonymize generated networks and use them for testing new software developed in open research facilities. The results indicate that modeling edge structure and the hierarchy together is essential for generating networks that are statistically similar but not identical to each other or the original network. In experiments, we generate many synthetic networks from two covert networks. Only a few structures of synthetics networks repeat, with the most stable ones shared by 18% of all synthetic networks making them strong candidates for the ground truth structure. The proposed generator can be used for many kinds of complex social networks that are either incomplete, or partially incorrect.

3.2.2 Introduction

Generating randomized synthetic networks is often used to facilitate the understanding of the network's properties, which depend on the network's structure [29]. The random network model introduced by Erdós and Rényi [25] generates edges between pairs of nodes in a network with a certain probability that is fixed for all edges. Such networks are highly random, as we expect. The randomness of such networks has sparked the interest of many researchers as to which properties of a network will be maintained, and which will be lost, when a given network is randomized. As more research was done, it was found that the random network model rarely arises in the real world.

A more advanced model that builds on the random network model, is the scale-free network model proposed by Barabási [30]. Many real-world social networks follow the scale-free network model. Another model is the Stochastic Block Model (SBM) [31], which is also an extension of the random network model. In the SBM nodes are grouped together, and the probability of an edge between any two nodes will depend on the probability of edge formation between the groups that the nodes belong to. The SBM produces community structures that arise in real-world social networks, but with one limitation of giving all nodes within a group the same probability of edge formation, which limits the difference in degree between the nodes present in a given group. To address this weakness, other variations of the SBM were proposed, such as the degree-planted SBM [32], and the degree-corrected SBM [33].

The Lancichinetti-Fortunato-Radicchi (LFR) benchmark [34] is a benchmark where synthetic networks of heterogeneity in the distribution of node degrees and community sizes are generated. The generated synthetic networks are customized using different parameters, such as the power-law exponents, which determine the distribution of node degrees, the distribution of community sizes, and the density of edges within and between communities, to mimic real-world networks. The LFR benchmark is often used to test the quality of community detection methods. Another model that generates synthetic networks by shuffling edges in real-life networks is presented in [35]. To make communities in the network more blurry we simply shuffle more edges, and usually the blurrier the communities are, the easier it is to spot stable communities, which are communities that remain stable through the blurring process. Another generative model for hierarchical multi-layer networks is presented in [36], and can be used to model networks with a hierarchical managerial structure.

The Hidden Community Detection (HICODE) model [37] identifies hidden communities by first performing a standard community detection method and then intentionally weakening the detected communities by removing or reducing the weight of edges. This allows weaker communities to be detected. Unlike our approach, the proposed method neither accounts for node hierarchy, nor preserves total edge connectivity in a rewired network. Another generative model creates hierarchical multi-layer networks that are often used to represent the hierarchical management structures, but does not consider reliability of a group or edge structure of the network [36].

In general, we are interested in on-line social networks that have been supported by the growing number of computer platforms and companies. Social networks create massive amounts of data that can help researchers study such networks and their properties. Such networks have been increasingly present with the increase in the amount of social networking platforms available. With that being said, access to such data is increasingly limited due to the many laws protecting the privacy of the users present in such networks. We are more interested in studying covert (hidden) networks, and for such networks it is even harder to collect data. Covert networks are naturally hidden, to avoid the detection of their members, and the illegal activities that they conduct. Collecting data on covert networks is very challenging, due to the secretive nature of the members of the networks. If such data was successfully collected, a very limited number of people would have access to it, due to the privacy laws put in place by many countries to protect its citizens from unjustified surveillance. Usually a small fraction of data becomes publicly available, when the members of such networks are prosecuted in court, and the transcripts of the investigation become

publicly available.

The main contribution of the network generator introduced here is creating synthetic networks that are structurally similar to real networks, but with anonymous nodes that are interconnected or clustered differently than nodes in the original network. The direct use of such anonymized networks is to provide a safe but statistically equivalent substitute for real networks for research on analytical tools for covert networks. A large number of such networks generated from a single covert network can serve as a set of alternative interpretations of the structure of that network. The distribution of frequencies of these structures enables analysts to quantify statistically expected outcomes of operations on the covert networks. A high frequency of presence of a particular synthetic structure among synthetic networks makes it a likely candidate for the ground truth structure.

3.2.3 Organizations Represented by Covert Networks

To collect data on covert networks, we need access to the flow of information between users of the network that are involved in the illegal activities of the network. The flow of information between users may be captured using wiretapped phone interactions, recorded conversations, or the discovery of written documents. For the rest of this section we will refer to such networks as covert networks, and to the users of such networks as members. We will also refer to members that are highly connected as a *group*, rather than the usual *community*, which is usually used to describe members that have a casual relationship, or *cluster*, which usually refers to members having a set of shared values or attributes. Covert organizations often have a hierarchical structure of management. The levels of hierarchy will vary depending on the size of the organization. In our proposed generator, we use parameters to describe the hierarchy level of a node, and thus separating the group leaders from the low-level nodes of the network. Given a real covert network, our generator will rewire proto-edges based on the groups, and the management hierarchy of the node. Having said this, our proposed generator combines two network models: the SBM for generating edges between groups, and hierarchy for maintaining the management structure of the network.

In small organizations, groups could be *independent* of the organizational structure of the network, but more often they have a *hierarchical* structure for management nodes. The number of hierarchy levels depends on the organization size. In covert networks, the hierarchical structure is important since law enforcement interdiction success vitally depends on correct recognition of the

leadership roles in a crime organization. For this reason, our model explicitly assigns to each node its place in the organizational hierarchy. In addition, each group is represented by two parameters, one defining proto-edge densities inside a group, and the other from this group to any other group. This extends also to densities of proto-edges from a group leader to its subordinates, and, separately, to other nodes. Given an original network, the generator individualizes it by randomly rewiring edges of its groups and its management hierarchy [36]. As a result, our model combines two network models: SBM for groups and hierarchical network for higher level management structure.

Most of the previously proposed synthetic network generators rely only on one network generative model. For example, in [38] the authors propose a network generator that creates synthetic multi-layered networks. A network generator presented in [39] implements small-world social networks with the desired high clustering coefficient, while [40] presents a generator of a dynamic scale free network with the given exponent γ with the minimal divergence from the scale-free model at each size of its evolution. Another network generator described in [41] creates dynamic networks with the prescribed group structure.

3.2.4 Data

The datasets used to test our proposed generator are the Caviar and Ciel datasets [42]. The Caviar dataset is based on a drug smuggling network, using data collected from 1994 to 1996 by an investigation of the West End Gang in Montreal, Canada. This gang mainly trafficked hashish and cocaine. What is unique to this dataset is that during the investigation, the police only confiscated shipment of drugs, but did not make any arrests until the investigation was over. The data consists of 11 2-months snapshots, where each snapshot showed the interactions between the gang members during that two-month period. Each snapshot of the network consisted of weighted proto-edges representing the calls made between nodes of the networks, and the frequency of those calls. The different snapshots allowed us to observe how the network reacted to the confiscations made by the police. Based on the publicly released court proceedings, we know which nodes of the Caviar network had management roles. From the proceedings, node N1 was identified as the leader of the hashish group, and node N12 as the leader of the cocaine groups. In Fig. 3.5, we show more of the nodes in managerial positions. It is important to note that using community detection algorithms, many low degree nodes were not assigned to any group [43].

We also test our network generator on the Ciel dataset [42], which is based on a drug trans-

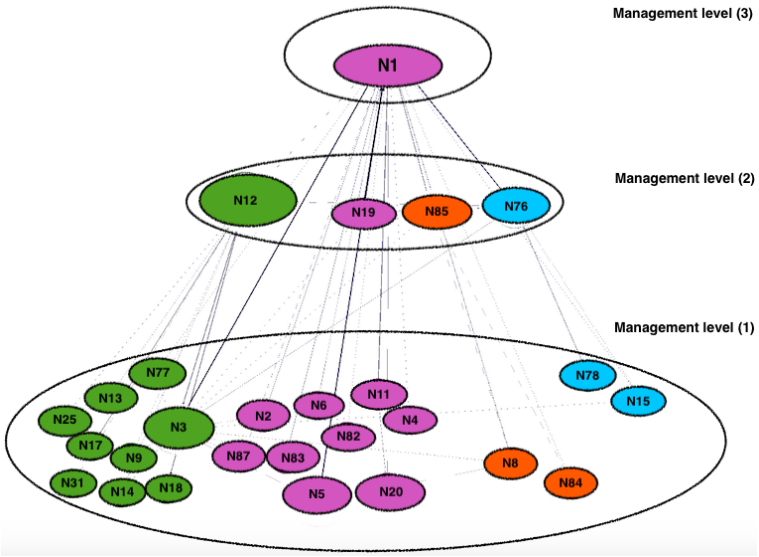

Figure 3.5: We present the combination of the stochastic block model (SBM) and the hierarchical multi-layer network model on the Caviar network. Same colored nodes belong to the same group, and nodes in the same hierarchical level have the same role in management of the network.

portation network that was mainly involved in trafficking hashish from Jamaica to Montreal, with data collected between May 1996 to June 1997. This dataset is both weighted and directed, with weights representing the frequency of node interactions, which was obtained from the phone calls surveillance records. High ranked members of the Ciel network were identified, with nodes N1, N2, and N10 being important members of the network. Just like the Caviar network, the Ciel network's ground truth was also not known. For both the Caviar and Ciel networks, the Louvain community detection algorithm [44] was used to find groups in the network. It is also important to note that we used an undirected version of Louvain community detection, due to the uncertainty of connectivity between the nodes in our network. For this, we simply transformed the directed edges of our networks into undirected ones, thus summing the weights of the directed edges between a pair of nodes, into a single weight of an undirected edge.

3.2.4.1 *Security vs Efficiency in Covert Networks*

Unlike social networks, covert networks cannot just maximize efficiency of the flow of information. The more efficient the information flows through a covert network, the less secure that network is. Therefore covert networks need to either prioritize efficiency or security, but not both. We used the relative betweenness centrality [45] to identify nodes of high importance in our

network, and thus nodes that belong in management positions. Betweenness centrality measures the centrality of a node in the network, by measuring the fraction of shortest paths of information flow that pass through the node. For all the results reported below, we used relative betweenness centrality, for ease of comparison, which is a normalized version of the betweenness centrality with its range being from [0,1]. We compare the Caviar and Ciel networks, as shown in Table 3.7, and it can be seen that Caviar prioritized efficiency, while Ciel prioritized security. In the Caviar network, node N1, who is the leader of the network, and nodes N3, N12, and N76, who are all in manager positions, have the highest relative betweenness centrality scores from all the nodes in the Caviar network. This shows that a lot of information flows through these nodes, which implies that a lot of nodes have access to them, which results in a high efficiency flow of information, but low security of the leader nodes. On the other hand, the leader of the Ciel network, node N10, has a very low relative betweenness, in comparison to the manager nodes N1, and N2. This shows that the leader was not easily accessible by many of the nodes in the network.

We chose to investigate both the Caviar and the Ciel datasets because they are very well studied [43], [46], [47], and they represent two types of covert networks, ones that prioritize efficiency, and ones that prioritize security. We show that using our generator we are able to generate synthetic networks that are similar to both networks, despite the fact that only partial knowledge of the ground truth was available for both networks. This is important because for many real covert networks the ground truth will not be known.

Table 3.7: The relative betweenness centrality scores of the management nodes in the Caviar and Ciel networks. Management nodes in the Caviar network had high betweenness scores, which made information flow in the network very efficient. In contrast, for the Ciel network, the leader node had a very low betweenness score, which shows that not many low level nodes had direct access to him, thus benefiting the security of the leader node, N10.

Network	Caviar				Ciel		
Node identifier	N1	N3	N12	N76	N1	N2	N10
RBC score	0.430	0.180	0.303	0.078	0.591	0.641	0.015
Rank of the score	1	3	2	4	2	1	7

3.2.5 BWRN Network Generator

Generating weighted networks has received a lot of attention over the last decade. Many of the proposed models aim to produce weighted networks that have properties similar to real social

networks. In [48]–[50], weighted network models, based on Barabási's preferential attachment model [30], are proposed with the goal of generating weighted networks with power-law degree distribution, positive degree correlation, and high clustering coefficient. In [40], a growth model that preserves the power-law degree distribution with nodes dynamically joining and leaving the network is proposed. Another model that generates weighted networks that meet user defined criteria, such as symmetry, clustering, and positive degree correlation, is proposed in [51]. Rather than user defined criteria, generated networks can be based on real input networks, to generate weighted networks that maintain the power-law degree, degree correlation, and high clustering coefficient of the real networks [52], [53]. In [54], Fortunato provides an authoritative review of the state of the art in community detection. The review covers many generative models and provides a more in-depth explanation of them for interested readers. It also demonstrates the poor performance of these models in replicating the community structure of the original network. All the discussed above models do not inform their rewiring processes about the hierarchy or group structure of the original network. Information about both of these aspects of a network enables the BWRN generator to rewire edges preserving internal connection densities within groups and between leaders and their subordinates. Hence, the synthetic networks created by the BWRN generator not only have an edge structure statistically similar to that of the original network, but also their group structures and organizational hierarchies are similar.

The process of running the BWRN generator on a given network requires two inputs. The first assigns to each node its place in the management hierarchy, and if a node has a management position, a list of its direct subordinates, and its membership in a group. Nodes with a management position may not have peers, in which case it will be considered a single node group. The second input assigns to each edge the identities of the groups to which its endpoints belong. This step creates proto-edges that are randomly assigned nodes from the relevant groups in the process of a synthetic network generation. The generated synthetic networks can then be used to analyze the stability of communities and hierarchical levels of the original network.

To further emphasize the final step of our process, we will focus more on the actual generating of synthetic networks. We will start with the model that serves as the foundation of our BWRN generator. Networks in general are weighted and directed, as is the case with the Caviar and Ciel networks. The weight of an edge between a pair of nodes is used to represent the intensity or frequency of interactions between that pair of nodes. A weight can represent the number of phone calls, meetings, messages, or any representation of an interaction occurring between two

nodes. Within our BWRN generator, we use the stochastic block model, as we mentioned earlier. Within the stochastic block model, we had some freedom to choose the model which will be used to assign weights to the generated edges. We provide the user with the option of choosing between one of two models to assign weights to edges, the first is the Weighted Random Graph (WRG) model [55], and the second is the *Bernoulli Weighted Random Network* (BWRN) model, which we introduce as an alternative to the WRG model. For the WRG model, let W denote the total sum of weights of all the proto-edges, and let E be the maximum number of proto-edges that can be generated between two blocks of nodes. The proto-edges are then generated using Bernoulli trials with probability $p = \frac{W}{W+E}$, and at the first failed trial, the trials stop. The weight of the proto-edge will then be equal to the number of successful trials occurring before the first failed trial. The proto-edge weights generated using the WRG model follow a geometric distribution.

Table 3.8: Notation table.

$B(n,p)$	Process of n Bernoulli trials with success probability p
d	Node degree
E	Number of directed edges
E^g	Maximum number of directed edges among nodes in group g
$E^{i,j}$	Number of directed edges from g_i to g_j
g_i	Group i
g_i^s, g_i^e	Groups with starting and ending nodes of edge i
$\lvert\neg i\rvert$	The number of all nodes not in a group g_i, and hierarchy $< s(i)$
k	Weight of generated edge
n	Number of nodes in the network
$ns(n,p)$	Random number of successes in $B(n,p)$
$n(d)$	Number of nodes with d degree
p_a	$w - p_B w_B$
p_B	Probability defining the variance of the generated weights
$s(i)$	Node directly superior to node i
w, w_i	Weight of a single edge, weight of edge i
w_B	$\lfloor w/p_B \rfloor$
W	Vector of w_i's of a weight of edge i
W_S, W_U	Sum of weights of edges, sum of unique weights of edges
$W^{i,j}$	Sum of weights of edges from g_i to g_j

We introduce an alternative model to the WRG, named *Bernoulli Weighted Random Network* (BWRN) model, see Algorithm 2 for pseudo code, and Table 3.8 for notation used. This model has two parameters, the first is a list of weights w's representing the weights of proto-edges present in the original network, and the second is probability p_B, which determines the variance of weight

distribution of the generated proto-edges. To generate proto-edges using BWRN, we start with the heaviest proto-edges, and move towards the smallest weighted proto-edges. For each of the weights stored in the list of weights w, an associated weight $w_B = \lfloor w/p_B \rfloor$ is computed. For each proto-edge with weight w in the original graph, a weight in the range of [0, $\lceil w/p_B \rceil$] is computed as the new weight of that proto-edge in the generated graph. To create edges in our generated graphs, we pick a pair of nodes that are not yet connected, and run w_B Bernoulli trials, with probability of success p_B. In the case that $p_B w_B < w$, we run one more Bernoulli trial, but now with probability $p_a = w - p_B w_B$. The probability of choosing weight k for a given edge, where $0 \leq k \leq \lceil w/p_B \rceil$ is defined as:

$$p_w(k, p_B) = \begin{cases} \frac{(w_B)!}{(w_B-k)!k!} p_B^k (1-p_B)^{w_B-k} & \text{if } k \leq w_B \\ w - p_B w_B & \text{for } k = w_B + 1 \text{ if } p_B w_B < w \end{cases} \tag{3.6}$$

The generated edges of a synthetically generated graph will have an average sum of weights that is equal to the sum of weights of the proto-edges in the original graph. The expected weight using Equation 3.6 is $p_B w_B + w - p_B w_B = w$. p_B defines the probability of a proto-edge of weight w not being generated, which is $(1-p_B)^{w_B}$ if $w = p_B w_B$ and $(1-p_B)^{w_B}(1 - w + p_B w)$ otherwise. As w and p_B increase, the probability of a proto-edge not being generated quickly decreases. For $p_B > 0.9$, for all proto-edges, even those with $w = 1$, their probability of not being generated is less than 1%. For all the experiments conducted we use $p_B = 0.875$, and for that value of p_B about 10% of the proto-edges with weight 1 in the original network will be lost, not appear in the generated network, but about 10% of another set of proto-edges will increase their weights from 1 to 2, thus strengthening the communities present in the network.

The variance in the distribution of weights generated for a given proto-edge with weight w is $w(1-p_B) + p_a(p_B - p_a) \approx w(1-p_B)$ [56], thus showing that the variance increases as w grows, but decreases as p_B grows. A large p_B will result in the generation of synthetic networks that are similar to the original network, and as we decrease the value of p_B the generated synthetic networks will become less and less similar to the original network. This feature of the model is essential for covert networks, which have both missing and incorrect edges (e.g., edges connecting a member of a Crime Organization (CO) to a person outside of it). Analysts involved in CO investigations can use an estimate of levels of this deception and select a value of p_B that would rewire enough edges to add the hidden connections and remove the incorrect ones, but not enough to disassemble the crime groups. For example, analysts can monitor whether the known CO members are losing

connections to other members, which would indicate that p_B is too low.

Procedure 2 BWRN model

Input: vector W; group g_i^s, g_i^e; int E, real $p_B \in (0,1]$;
function $ns(n,p)$; **process** $B(n,p)$;
Comment: all symbols are defined in Table 3.8
Sort proto-edges in the order of groups they connect and their descending weights
  for i=1 to E **do**
    $w_B = \lfloor w_i / p_B \rfloor$
    randomly select a pair of not yet connected nodes in groups g_i^s, g_i^e
    w_i^g = ns(w_B, p_B)
    if $p_B w_B < w$ **then** increase w_i^g by ns(1, $w - p_B w_B$)
    end if
  end for

3.2.5.1 *Group Detection in Generated Networks*

To test the quality of our generated synthetic networks, we compare the communities detected on our synthetic networks, to the communities detected on the original network. For all the community detection conducted in our experiments, we used the Louvain community detection algorithm [44], to detect groups in our generated networks, and compare those groups to the groups detected in the original network. There are many other alternative community detection methods, such as Speakeasy [57] and Natural Communities [58] which we used and yielded similar results to Louvain, CPM [59], modularity maximization [60], adaptive [61], or fast modularity [62]. The Louvain method partitions the network into communities that maximize modularity. First, it finds small communities that optimize modularity locally on all nodes. This process is repeated until the modularity of the whole network is optimized. Speakeasy is a label propagation clustering algorithm that can detect both overlapping and non-overlapping communities, by allowing nodes to join communities based on an exchange of labels between connected nodes. Natural communities method is based on the local optimization of a fitness function by performing a local exploration of the network searching for the natural community of each node, while allowing a node to belong to more than one community. This allows overlapping communities to be detected. Users of our BWRN generator have the freedom to choose any of the mentioned, or other, community detection methods to test the accuracy of the generated networks.

We utilized the Louvain community detection method on the Caviar and Ciel networks. It was interesting to find that the detected groups did not always match with what we expected. There

are several reasons as to why that may have been the case. First of all, the Caviar and Ciel networks are partially incomplete, and may have some undetected edges. Second, and more importantly, high degree nodes are so often very well connected to many groups of the network, thus resulting in those nodes not truly belonging to a single group of the network. For example, in the Caviar network, we know that nodes N1 and N3 belong to the same group, but since N3 is very well connected to other groups of the network, the generator may sometimes assign N3 to those other groups. This is not the case with just N3, other well-connected nodes are often misclassified too. Well-connected nodes are often connected to many different groups of the network, considering that they play some role in each of the groups that they are connected to, thus signaling that this node plays various roles within the network. As we continued to generate more synthetic networks, we observed how the roles of certain nodes changed. This helped us further understand the nature of operations being conducted in the original covert network. We also look into the hierarchy levels of the network, and try to detect different hierarchy levels to reveal the hierarchical structure of the network. We use relative betweenness centrality as a hierarchy measure, and it has been shown that in networks there is a strong correlation between the betweenness scores and the hierarchical levels of nodes [63]. We find the nodes with the highest relative betweenness in the generated networks and compare them to those of the original network, to measure how well the leadership hierarchy (leader nodes) was preserved in the generated networks. A Combined Score (CS) measures the overall similarity of generated networks to the original one, by taking the product of the Group NMI Score (GS) and Jaccard Leadership Score (LS), as below.

$$\mathrm{CS} = \mathrm{GS} * \mathrm{LS} \tag{3.7}$$

3.2.5.2 *Generating Synthetic Networks*

To use the BWRN generator, the user needs to do the following. First, anonymize their data by removing any personal information, and providing the generator with a list of nodes, and the management hierarchy that each node belongs to. Second, the user has to provide the list of all proto-edges present in their network, and the weight of each proto-edge. This list of proto-edges should be composed of a source node, a target node, and the weight of the proto-edge connecting them. Third, the user should provide a list of groups present in the network. This list can either represent the ground truth groups of the network, or can simply be the list of groups detected on the original network, using any group detection method, as we mentioned earlier. Once these

are provided by the user, we can start processing the input data. We start with computing the probabilities of proto-edge connections between groups of the generated network. As mentioned earlier, we use one of the two models to assign weights to the generated weighted edges. Both models will separate the proto-edges in several classes, and generate weights and edges for each class separately. The first class will represent the internal proto-edges of a group g_i of size $|g_i|$, with members all belonging to the same hierarchy level. For such group g_i, there are $E^i = |g_i|(|g_i| - 1)$ directed proto-edges, with a sum of weights denoted by W^i. The second class will include proto-edges across nodes belonging to different groups, i.e. between g_i and g_j. In this case, there are $E^{i,j} = |g_i||g_j|$ proto-edges from g_i to g_j and $E^{j,i}$ proto-edges from g_j to g_i, with the sum of their weights denoted by $W^{i,j}$ and $W^{j,i}$ respectively. The third class will include proto-edges between a superior node $s(i)$ and the members of its group g_i. In this case, there are $E^{s(i),i} = E^{i,s(i)} = |g_i|$ proto-edges going in each direction, from $s(i)$ to g_i and from g_i to $s(i)$, with the sum of their weights denoted by $W^{s(i),i}$, and $W^{i,s(i)}$ respectively. The fourth, and final, class will include the set of proto-edges from a superior node $s(i)$ to members not in its groups, so members not in g_i, such members will be denoted by $|\neg i|$. Here there are $E^{s(i),\neg i} = |\neg i|$ proto-edges in each direction, with the sum of their weights denoted by $W^{s(i),\neg i}$.

The two models used to assign weights to the generated edges are the Weighted Random Graph (WRG) model [55], and the Bernoulli Weighted Random Network (BWRN) model. Both models are discussed in depth in Section 3.2.5.

Our BWRN generator allows for the assignment of an arbitrary number of management levels, but given that the Caviar and Ciel networks are relatively small, we set the limit of the hierarchy levels to three, for the conducted experiments. Hierarchy levels, higher than the lowest hierarchy level, can have a varying number of members, depending on the total number of members present in the immediately lower level. For the three levels of management used in our experiments, the third management level consisted of the leader node of the network, which is referred to as *boss*. The second hierarchy level will be composed of intermediary nodes, referred to as *managers*, that will connect the *boss* node with the rest of the network. The first level of hierarchy consists of the remaining nodes, and is composed of several groups. Nodes residing in the first level of hierarchy are referred to as *members*. The BWRN generator will attempt to detect the leader and manager nodes automatically without any extra help from the user. When we were deciding how many managers should we assign per member, we referred to the literature and found that small companies tend to have an average of four employees per manager [64]. Given the differences

between a covert network, and a small company, we decided to go with six members for each manager, with the reasoning that in covert networks we want to limit the number of members that have direct access to the boss node.

Hierarchy is not always consistent across all groups of the network, some groups may have more levels of hierarchy than others, while other groups may have no hierarchy at all. Groups with no hierarchical management are referred to as *independent*. Groups that consist of a small number of nodes, or with a very few number of edges are likely to be independent. One example of an independent group is the money laundering group in the Caviar network, with members only having outgoing edges to nodes outside the groups, and no incoming edges.

3.2.5.3 *Baseline Generator*

To test the accuracy of our BWRN generator, we needed a baseline generator to compare it against. Initially, our baseline generator was the Erdós Rényi [25] random network generator, but we had to use a better baseline. Our new baseline generator is an extended version of the stochastic block model that supports weighted and directed edges. The SBM baseline generator takes as input a list of the nodes present in the network, a list of the network edges, and a list of the groups detected in the original network. We then proceed to do the following for each group, we sum the weights of all the edges within the group, and the weights of all the edges between this group, and other groups in the network. We then use these sums to calculate probabilities of edges within the groups, and between groups. We use the built-in numpy random choice method [65], to pick a pair of groups within our network. We then choose a source node from the source group, and a target node from the target group. For edges within a group, we need to make sure there are no self loops, thus the source and target node must not be the same node. Once a pair of nodes is found, we execute a Bernoulli trial with probability based on the calculated probability of edge formation between the source and target group. If the Bernoulli trial succeeds, we increase the weight of the edge between the pair by one. We repeat this process until we generate the same total weight of edges between the chosen groups, as present in the original network. Once we are done processing all the groups, we should have the same total weight of edges in the generated network, as present in the original network. As mentioned for the BWRN generated network, we once again use Louvain community detection to detect groups in the baseline generated networks.

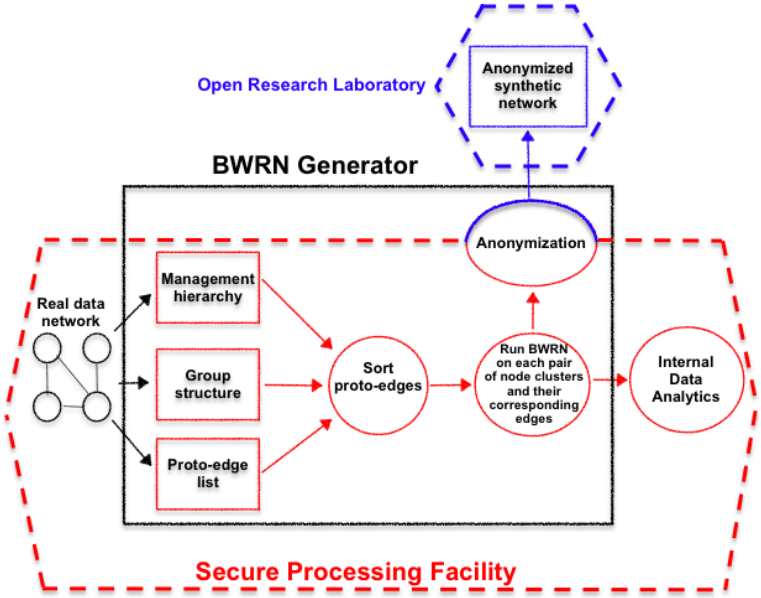

Figure 3.6: The graphic framework showing the process of generating networks using the proposed generator. Circles represent the processing stages of the generator, rectangles stand for datasets, and hexagons show computational centers. The red color denotes secure processes and data with access limited to within the secure processing facility, while the blue color denotes open access data and facilities.

3.2.6 Flowchart and Time Complexity of the BWRN Generator

Fig. 3.6 summarizes the process of generating synthetic networks using the proposed generator. Although we start with a single real network, the synthetic networks generated from it will all be unique. Some generated networks may have the same group structure, but they will all have a unique set of generated edges, so after node anonymization, it would be difficult to recover the original input network from the single synthetic network. Thus, the anonymized synthetic networks can be shared with researchers that operate in open research laboratories for testing and validating software on networks that are similar to real covert networks.

The time complexity of executing the BWRN generator presented in Fig. 3.6 can be established as follows. The input data consists of the management hierarchy, group structure, and the list of proto-edges of the network to be rewired and may be already included in the data associated with the real network to be rewired. Alternatively, the management hierarchy can be uncovered using the relative betweenness centrality with complexity of $O(n(n\log n+E)$. This is by factor of n the most computationally intensive part of the BWRN generator execution, but it can be avoided if ground truth about the management hierarchy is uncovered by investigation. The group struc-

ture could be found by Louvain (or any other community detection) algorithm with complexity $O(n \log n)$.

The network rewiring process consists of two main steps. The first sorts the network proto-edges according to groups they connect and in descending order of their weights. The complexity of sorting is $O(E \log E)$. The second step runs the BWRN generator, on every proto-edge of the rewired network. Since each proto-edge is rewired once, the complexity of this step is $O(E + \sum_{i \in W} \log(w_i))$, where w_i denotes the weight of edge i.

3.2.7 Results

To compare the quality of the synthetic networks generated using the BWRN generator we used Louvain community detection to detect groups in the generated networks. We then used the Normalized Mutual Information (NMI) method [66] to measure the similarity between the generated groups, and the original groups detected in the original network. To avoid any statistically insignificant outliers, we generated 100 synthetic networks and took the average scores.

We do not just measure how similar the detected groups of the generated networks are to the original groups of the original networks, we also measure how well the management levels were maintained in the generated networks. As we mentioned earlier, to measure how similar the groups of the generated networks and the original networks are, we used Louvain community detection and Normalized Mutual Information. To measure how well we maintained hierarchy and management levels, we detect management nodes in the generated networks, by finding the nodes with the highest relative betweenness centrality. To clarify, assume the original network has n management nodes, all with known relative betweenness scores. We then proceed with finding the top n nodes with the highest relative betweenness scores in the generated networks, and use a 90% threshold of the relative betweenness centrality scores between the top nodes in the generated network, and the top nodes in the original network. All the tests conducted are based on the Caviar and Ciel datasets, and for both datasets we create three sets of synthetic networks. We used Bernoulli Weighted Random Network (BWRN) method to create the first set, Weighted Random Graph (WRG) method to create the second set, and the weighted SBM (baseline generator) to create the last set.

In Table 3.9 we see results solely based on the similarity of group structures between the generated networks and the original networks. Results are reported for both the Caviar and Ciel datasets, and are based on the three sets of synthetic networks mentioned earlier. We can see that

the groups detected on networks generated using BWRN method are by a factor of two or more similar to the groups in the original network, when compared to the WRG method. The baseline SBM and the BWRN have similar performance.

Table 3.9: Results show NMI scores from comparing the groups in networks generated from the Caviar and Ciel networks to the groups in the original networks. The results show performance of BWRN generator, and generators using the Weighted Random Graph model and the weighted SBM baseline model.

Original Network	Caviar			Ciel		
Generator	Generator		Weighted	Generator		Weighted
	BWRN	WRG	SBM	BWRN	WRG	SBM
mean	0.815	0.356	**0.850**	**0.800**	0.513	0.761
median	**0.839**	0.365	0.739	**0.883**	0.567	0.751
min	0.415	0.088	0.358	0.734	0.288	0.551
max	0.953	0.565	0.883	1.000	0.692	1.000

By measuring just the quality of the generated groups, it is hard to differentiate between the BWRN model and the SBM weighted model. In table 3.10 we demonstrate the importance of leadership detection. We use Jaccard metric [67] to measure the similarity in management levels between the generated networks and the original networks. The results show that our BWRN generator using the BWRN method preserved the hierarchical leadership structure two times better than the SBM weighted baseline generator, and almost four times better than the BWRN with the WRG method. This shows that the BWRN with BWRN does a much better job preserving the leadership of networks than other methods, and also performs equally or slightly better than other methods at maintaining groups.

Table 3.10: The first row of the results shows group structure similarity from Table 3.9, the second the leadership similarity, and the last row shows the combined Score that is the product of the first two scores. In all three rows, the best score is shown in bold font.

Metric	BWRN	WRG	Weighted SBM
Group NMI median (GS)	**0.839**	0.365	0.739
Jaccard Leadership (LS)	**0.681**	0.402	0.308
Combined Score (CS)	**0.571**	0.146	0.281

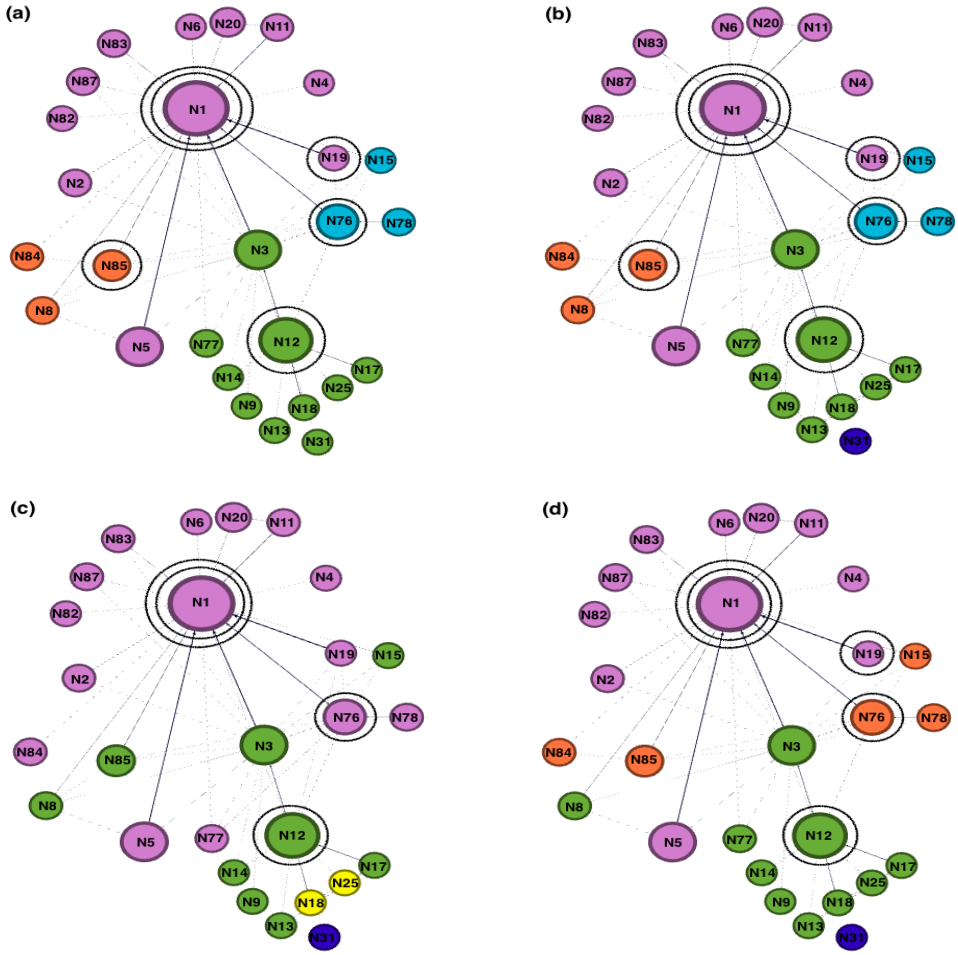

Figure 3.7: The presented Caviar networks depict a) the groups in the original network, and groups detected in the synthetic networks with similarity that is (b) highest, (c) lowest, and (d) average.

3.2.7.1 *Extension Methods*

With the achieved set of scores, we were interested to see if we can further improve our generator by trying out different edge addition methods. Thus, we tried three different methods. The first method adds edges based on the number of common neighbors between a pair of nodes. The second method adds edges using the triangle closing method. Finally, the third method adds edges using preferential attachment.

The approach for edge additions based on common neighbors is as follows. All pairs of nodes (x,y) are assigned a score based on the number of neighbors that they have in common, such that $score(x,y) = \Gamma(x) \cap \Gamma(y)$, where $\Gamma(x)$ denotes the set of neighbors of node x, and $\Gamma(y)$ denotes the set of neighbors of node y [68]. The pairs with the highest scores will be assigned an edge in the generated network, until we assign the required number of edges. The first variation is based

on assigning each pair of nodes an edge of weight 1. The second variation sets the weight of the added edge equal to the common neighbors similarity $score(x,y)$. For both variations, if a pair of nodes already share an edge, the weight of their edge will be increased by the given amount.

The triangle closing method, presented in [69], is based on the idea of adding edges locally within a network to connect nodes with common neighbors. The implementation of the triangle closing method is as follows. First, we pick a node y and then choose two of its neighbors, nodes x and z. Next we add an edge between x and z, thus closing the triangle of nodes (x,y,z). We extend this method by adding some conditions to choosing the neighboring nodes. Each neighboring node to y will be assigned a score based on whether it is in the same group or management level as node y. Nodes in the same group and management level will obtain a higher score. The nodes x and z with the highest scores will be selected to add an edge across. We ran tests on the triangle closing method both with and without the attribute score extension.

Our final extension method is based on preferential attachment, which has been widely used to generate edges in networks, as presented in [68] and [30]. Each pair of nodes (x,y), will be assigned a score based on their degree. The score is defined as $score(x,y) = deg(x) * deg(y)$, where $deg(x)$ denotes the degree of node x, and $deg(y)$ denotes the degree of node y.

To test these extension methods, we generated 100 networks with 70%, 80%, and 90% of the edges generated using our generator. The remaining edges were added using each of the three extension methods. We used Louvain community detection on the resulting networks, and used NMI scores to compare the detected groups to the original groups, but found that the proposed extension methods only worsened our results.

3.2.8 Stability of Generated Structures

Moving away from how similar the generated networks are to the original network, we are now interested in how stable the generated networks are. The generated networks are statistically similar to the original network, with some small perturbations to the proto-edges present in the original network. As we compare the structures of the generated networks, we can find the most stable structure, as it will be the structure most present across the generated networks. To find whether our original network's structure is stable or not, we can see how many of the generated networks share the same group structure as that of the original network. If only a few of the generated networks are structurally similar to the original network, we can conclude that the structure of the original network is not stable. On the other hand if many of the generated networks are

structurally similar to the original network, then we will know that the original network's structure is stable, and is resistant to the perturbations caused by the generator.

We apply this analysis to the Caviar network by generating 1000 synthetic networks. We compare the structure of those synthetic networks to each other, and to the original network. Our analysis is as follows, first we create a meta-graph G, with nodes representing the generated networks, so our graph G will consist of 1000 nodes. Nodes in G will be connected by an unweighted and undirected edge if they share a matching group structure. We will have two versions of this meta-graph G, the first version will have edges representing exact matches, and the second will have edges representing flexible matches. In a flexible match, it is allowed for up to one node difference in each group present between the two networks. In Figure 3.8, we show the results for exact matches in meta-graph G. It is important to note that the original network's structure only had ten exact matches among the 1000 generated networks, thus showing that the original structure is not very stable. The original structure is sensitive to any small perturbation in node connectivity. We find that the most stable structure occurs 177 times with exact matching, and 301 times with flexible matching. From the top ten most stable structures, each appear at least 20 times for exact matching, and at least 206 times for flexible matching. All the other generated networks only had a few matches for both exact and flexible matching.

The challenge in finding exact and flexible matching was mainly how to do it efficiently. Comparing the groups of every generated network to the groups of every other generated network can be very time-consuming, with a total of $1000 * 1000$ total comparisons. To make this process efficient, we do the following. First, we sort each network's detected groups in decreasing order of size, and within each group, nodes will also be sorted in decreasing order of their identifiers. This enables us to proceed with comparing the detected groups of the generated networks. To minimize the number of comparisons needed, there are several checks that can confirm that a pair of networks will not have a match. If two networks have a different number of groups, they can never have a match. Assuming that two networks do have the same number of groups, we then proceed with comparing the size of the groups present in the two networks. Since the groups have been sorted, the networks may have a match only if all their groups match in size, otherwise they will not have a match. Only if two networks pass all the previous checks, do we then manually compare their groups to check whether these two networks are in fact an actual match. This process makes the exact and flexible matching very efficient, which is a necessity considering the number of networks that we are comparing against each other. See Algorithm 3 for the exact matching

algorithm pseudo code.

Algorithm 3 Exact Matching Algorithm

Step 1: Sort the detected communities of each network in decreasing order of size.
Step 2: Sort each detected community for each network based on node IDs.

```
for each network structure ns1 do
    Add ns1 to list of visited network structures
    for each network structure ns2 do
        if ns2 in list of visited network structures then
            Break
        end if
        if number of communities in ns1 == number of communities in ns2 then
            Match = True
            for i in range number of communities in ns1 do
                if len(ns1[i]) != len(ns2[i]) then
                    Match = False
                end if
            end for
            if Match == True then
                for i in range number of communities in ns1 do
                    S = intersection(ns1[i], ns2[i])
                    U = union(ns1[i], ns2[i])
                    if abs(len(S) - len(U)) > 0 then
                        Match = False
                    end if
                end for
                if Match == True then
                    Add ns2 to ns1's list of matching structures
                    Add ns1 to ns2's list of matching structures
                end if
            end if
        end if
    end for
end for
```

Identifying stable groups in criminal networks is important. It can help analysts concentrate on group structures that are occurring most frequently, and thus represent plausible interpretations of the data based on those stable networks. Stability measures can also be of use for simulating interdiction on many alternative structures of the network, to reveal a wide range of outcomes. A distribution of the interdiction outcomes for the original network can be computed using probabilities of the outcomes of the generated networks, based on the number of occurrences present of

these networks in the meta-graph.

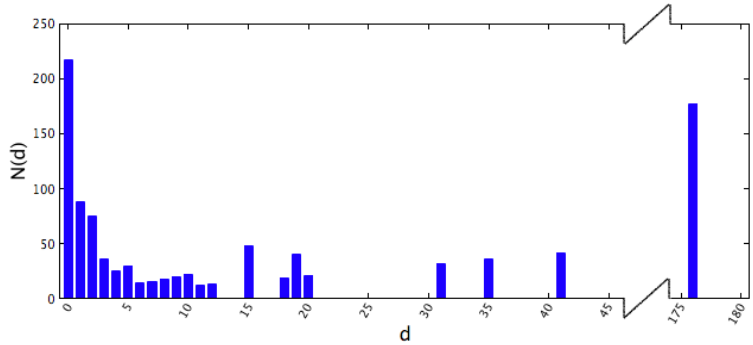

Figure 3.8: Histogram of the meta-graph degree distribution for exact matches between the generated networks. The x-axis is the node degree d, and the y-axis shows the number of nodes with d degree n(d).

3.2.9 Conclusion and Discussion

In this section, we introduce and make available a synthetic network generator that produces statistically similar networks to a given real or synthetic network. This generator will allow researchers to share data, such data will be synthetic, but it will be a representation of a realistic network. These networks can then be shared amongst researchers who need realistic data to develop and study tools for network analytics. Owners of private data can protect their data by providing the generator with abstract structural information, while maintaining the personal and operational information of the network private. To do this, a user would have to provide our generator with three lists. This includes a list of nodes, a list of groups and management roles, and a list of node groups as well as the statistics about edges and weights across and within those groups. With the mentioned lists provided, the original network provided is succinct and void of any personal data. Our generator also allows for further shuffling or randomization of node IDs to further maintain the anonymity of any personal information present in the original network. The generated networks are of the format source node ID, target node ID, weight, which is also the same format expected for any input network provided to the generator. We introduce the Bernoulli Weighted Random Network method that creates networks with an average total sum of weights equal to the original network, and we use the Stochastic Block Model to generate alternative group structures to those in the original network. We use a combination of both the SBM and a hierarchical network model, to preserve the hierarchical aspects of the original network.

The proposed BWRN generator was tested on covert networks, but is equally applicable to social networks. This generator aims at aiding the analysis of covert networks with hidden or incomplete information about their nodes, edges, or internal structures. This generator can enable researchers to further study covert networks, with multiple versions of the same network, and can help researchers develop algorithms on synthetic networks that can be used in real-life applications and on real networks.

The second application of this generator is to be used as a tool for analysts, to utilize it for network analytics on the original network. Studying the stability of the original network, by comparing its structure to the structure of the generated networks, can enable analysts to better understand the original network at hand. The more synthetic networks we have, the more we can study and understand the nature of the operations occurring in the original network. Community structures that are frequent can be studied to understand what makes them stable despite the perturbations caused by the generator.

We tested the BWRN generator on two real covert networks, the Caviar and Ciel datasets. To measure the quality of the generated networks, we used the Louvain community detection algorithm, to measure the similarity between the communities of the generated networks and the communities of the original network. After detecting groups using Louvain, we used the Normalized Mutual Information metric, and the Jaccard metric, to measure the group and hierarchical similarity between the generated networks and the original network. Our results demonstrate that the generated networks and the corresponding original network were highly similar. From our results, we concluded that accounting for groups, as well as management hierarchy of a network, is essential for generating synthetic networks that are statistically similar to the original network.

For future work, we plan to extend this generator by adding a fourth step that allows us to scale the generated networks. One way to do this is to replicate parts of the original network, and to add additional levels of management. Expanding the generated networks will allow researchers and analysts to study large and complex criminal networks, which are hard to find real data on.

CHAPTER 4
MINIMIZING UNCERTAINTY COST OF NETWORK STRUCTURE FROM NOISY DATA

4.1 Overview

Community structures form a basis of many network analytic tools. Yet, the prevalence of noise in data massively collected for large networks and intentionally obfuscated in covert networks distorts edges and community structures of such networks. To address this challenge, we repeatedly rewire a given network with a community structure created from the noisy data while controlling the standard deviation of the edges' weights and connectivity. This process creates a large set of the original network variants compatible with the noisy data. We use several entropy-based metrics to identify the lowest uncertainty community structure among them. However, in many applications, e.g., maintaining a complex system or disrupting a criminal or terrorist organization, the uncertainty cost is more essential than uncertainty itself. Hence, we introduce a novel heuristic that creates a community structure with the lowest expected cost of uncertainty. This cost can be defined by the user as a function of differences between two community structures. To validate our approach, we apply it to two criminal networks, one terrorist network and one social network. The results demonstrate that our approach creates a set of feasible variants of an original network, finds for it the minimum uncertainty community structure, and creates a community structure that minimizes the expected cost of uncertainty.

4.2 Introduction

Increasingly, network science research involves real-world networks defined by the data collected about them. Collection processes often involve continuous monitoring of a large number of nodes and a vast volume of interactions resulting in massive data sets. However, such collection is often marred by errors and mislabeling of nodes and interactions, resulting in noisy data [70],[71]. The causes of such noise can be classified into several categories of which we discussed three here. The first one arises when the monitored relations are not directly observable, so collected data contains proxy relations that only indicate probable presence of these desired ones. In the context of social networks, an example could be a trust between two people, that usually implies frequent

Portions of this chapter have been included in: A. Mandviwalla, A. Elsisy, M.S. Attique, K. Kuzmin, C. Gaiteri, and B. K. Szymanski, "Network Analytics Enabled by Generating a Pool of Network Variants from Noisy Data," *Entropy*, 25(8), 2023. https://doi.org/10.3390/e25081118, Featured paper in vol. **25,** issue 8.

communication between them. Hence, repeated calls between two cell phones often are treated as a sign of a trust between their owners, while in reality some of these calls could be made by people different than the owners, and the calls could be strictly professional and void of any personal interactions between the callers. Likewise, a common source of data noise in protein-protein interactions (PPI) networks is the current inability of monitoring technology to observe PPI directly. Hence, that data contains all proteins at the site of a reaction, yielding false positive interaction between some protein pairs [72]. In covert networks, another category arises through intentional hiding by members of their involvement and interactions by avoiding communicating within the network using easy to trace means, such as cell phones with registered ownership [43], [73]. Such members may also use wiretapped phones only for private communications [74]. In many networks with massive data collections, the third category of data noise arises because of the presence of a low but persistent rate of erroneous experimental measurements that distort the valid results [75], [76].

The presence of noise in network data distorts the detection of the network's edges and likely modifies the network's community structure. To address the challenge of finding edge and community structures in real-world network's noisy data, the authors of [15] introduce the Bernoulli Weighted Random Network (BWRN) generator. Given the basic parameters of a network recovered from noisy data, such as the lists of nodes, weighted degrees of all nodes, communities, and the hierarchy, the generator produces a set of networks randomly generated with the given parameters. Each network in this set is statistically equivalent to all others, but with a modified edge structure and, consequently, also community structure. Effectively, this generator rewires networks using a combination of the Stochastic Block Model (SBM) [31], to limit the changes to the generated network's community structure and a hierarchical model, to preserve the network's hierarchy (other generative models used to create networks with communities are discussed in [77]). The extent of rewiring is controlled by the user-provided parameter, $p_B \in (0,1]$, that defines the variance of the generated weights distribution. As p_B approaches 1, the rewired networks become more similar to the real network and to each other.

In this paper, we introduce the methods for finding the most likely community structure from the noisy network data, assessing the probability that this data structure agrees with the ground truth and estimating the level of uncertainty about the assignment of nodes to communities. Using the BWRN method, we generate a set of r networks from the given noisy network data and find the non-overlapping communities in each network. Then, we cluster these networks into

$s \leq r$ groups, each containing networks with the same community structure. Since the network community structures are robust to minor edge perturbations [78], for larger r, $s < r$ and the ratio of size of each cluster to r approximates the probability that the corresponding structure is a ground truth.

Entropy-based metrics were used in networks for different purposes. In [79], the authors measure the uncertainty of nodes considered for strengthening or weakening their existing links with their neighbors. In [80], entropy-based metric measures the vulnerability of communities in complex networks. Similar metrics have also been used to measure the structure similarity of nodes in complex networks, based on the node's local structure topology [81]. Entropy metrics have also been used to measure evolution of human communication in [82]. The authors found that Shannon's entropy generally decays with time when the social network stagnates with the same members. In [83] the authors use node attributes to define the affinity of nodes to belong to the same community and extend SBM to quantify uncertainty in the node's community membership. Their approach differs significantly from ours. We use the network edge structure to define the affinity of nodes to be in the same community and we rewire edges in a network to create alternative community structures to the one in the network being rewired.

To quantify the community structure limits of uncertainty and predictability, we introduce a *set* entropy-based metrics that are adapted from human mobility entropy models proposed in [84]. We apply these metrics to the set of s community structures derived from r generated networks. The original three metrics were introduced in [84] to quantify the limit of human mobility predictability. The precision of localization was low, measured in square miles, as it was defined at each instance by the area serviced by a cell tower that transferred the current call of the given user on each trip. We make these metrics useful for our problem by mapping each node into a cell tower in the human mobility problem, and each community with a node n into visits of this node to all members (towers) of its communities in all community structures. We also use as a metric the classical Shannon entropy [85] that measures the uncertainty of the outcome in a given experiment based on the probability distribution of all possible outcomes.

In our application, the entropy-based metrics are used to measure the uncertainty present in each community structure. The goal is to construct a community structure with the lowest uncertainty and use it as the most likely ground truth structure for the given network data. Finally, we present a new heuristic to create the community structure that minimizes the expected cost from uncertainty of the data. Such structures are important in the investigation of criminal organizations,

and in planning disruptions of such organizations.

The rest of the paper is organized as follows. Section 4.3 describes the methodology used in our paper. In Section 4.4.2, we present the data sets used for validation. The design of validation and their results are presented in Section 4.4.3, and the conclusions are discussed in Section 4.5.

4.3 Methods

In this section, we present preprocessing of the noisy network data, three novel entropy-based metrics, and a new heuristic that given a network constructs the community structure that minimizes the expected cost arising from the noisiness of network data.

4.3.1 Noisy Network Data Preprocessing and the Shannon Entropy Metric

Given a network with a set N of nodes, denoted as $\{n_1, n_2, \ldots, n_{|N|}\}$, and a set $E \subseteq N \times N$ edges, we use the aforementioned BWRN generator to rewire the given network r times, creating a set of r networks statistically similar to each other. Then, we use the Louvain community detection algorithm [44] to detect non-overlapping communities in each generated network. We cluster these networks putting in each cluster those with the same community structure (thus the same communities), C_i, for $i = 1, \ldots s$, where s stands for the number of created clusters. The number of networks in a structure C_i is denoted as w_i. This set of s communities structures constitutes the feasible ground truth structures for the given noisy network data. When r tends to infinity, a fraction $f_i^C = w_i/r$ converges to the probability of structure C_i being the ground truth for the given noisy network data.

This preprocessing is dependent on the choices of its parameters: p_B, which defines the extent of rewiring, and r that defines the number of generated networks. Clearly, the smaller p_B is, the larger r must be, since stronger rewiring requires more networks generated to create all the feasible community structures. Since the presented method is a heuristic, to obtain the best results, experimenting with some ranges of these parameters is recommended.

The first proposed entropy-based metric is the classic Shannon entropy computed over an entire set of community structures by setting $p_i^C = f_i^C$. The corresponding equation is

$$e_s^C = -\sum_{i=1}^{s} p_i^C \ln p_i^C \tag{4.1}$$

It is clear from the definition that the most reliable ground truth structure is the community structure C_i with the highest fraction f_i^C.

4.3.2 Set Entropy-based Metrics

In this section, we introduce *set* entropy-based metrics adapted from the mobility entropy metrics proposed in [84]. The authors use towers that serviced the calls of each user as the approximate location of the user and define three entropy measures for increasingly complex mobility patterns of towers servicing the calls. The first is the random entropy, which accounts for the number of distinct locations (towers) V_i, visited by user i. It is defined as $S_i^{rand} = log_2 V_i$. The second is the temporal-uncorrelated entropy, which accounts for the historical probability $p_i(j)$ that a location j was visited by user i. It is defined as $S_i^{unc} = -\sum_{j=1}^{V_i} p_i(j)\, log_2 p_i(j)$. The third and most complex entropy accounts for the frequency and order of visitations made by each user. Let $T_i = X_1, X_2, ..., X_L$ denote the sequence of location at which user i was observed at each consecutive hourly interval. In this case, the real entropy is used. It is defined as $S_i = -\sum_{T_i' \subset T_i} P(T_i')\, log_2[P(T_i')]$, where $P(T_i')$ is the probability of finding a particular time-ordered subsequence T_i' in the trajectory T_i. The authors also introduce important measures of predictability $\Pi \leq \Pi^{max}(S,V)$, where Π^{max} represents the limit of predictability for each user, It is defined as $S = H(\Pi^{max}) + (1-\Pi^{max}) log_2(V-1)$, where the binary entropy function $H(\Pi^{max}) = -\Pi^{max} log_2(\Pi^{max}) - (1-\Pi^{max}) log_2(1-\Pi^{max})$. Maximal limits of predictability for Π^{rand} and Π^{unc} were also determined and extracted from S^{rand} and S^{unc}, respectively.

To apply those entropy measures designed for the human mobility model to our community structures uncertainty model, we use the following two mappings. The first mapping, as shown in Fig. 4.1, is for the random and temporal-uncorrelated entropy metrics. In this mapping, we project each unique community i across the entire set of community structures in our model onto the unique location V_i in the mobility model.

For the real entropy, S, we map, as shown in Fig. 4.2, each node $n_i \in N$ onto the unique location V_i. Each community structure, C_t, is mapped onto time slot t in the mobility model. Node n_i, corresponding to the unique user i in mobility model, is mapped onto a list of members in its community C_t. These members correspond to locations visited by user i in time slot t in mobility model and are listed in the decreasing order of their frequency of pairings with node n_i. In other words, node n_i will first list its community member n_j that most frequently appears with n_i in the same communities across all r networks.

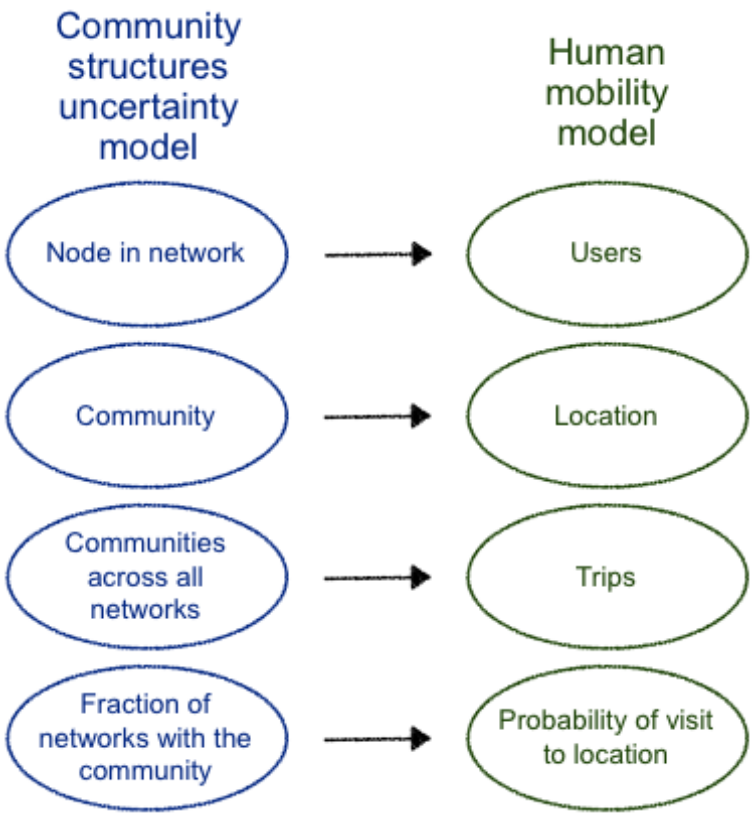

Figure 4.1: Mapping the node's community structure uncertainty model onto the random and temporal-uncorrelated entropy metrics in the human mobility model.

4.3.3 Beyond Entropy and Edges: Constructing the Community Structure Minimizing the Cost of Uncertainty About Community Memberships

So far, we used the entropy measures to find the community structure C^{opt} with highest probability to be a ground truth structure and therefore having the lowest entropy. Doing so we were limited to communities found by Louvain community detection relaying on edge structure. Yet, we want to construct a new structure that ensures the lowest expected cost arising from uncertainty about communities of this structure. Such communities may not directly appear in any rewired network structure. This freedom may enable this community structure to lower Shannon entropy against the set of networks rewired from the hybrid network created by merging edge structure of the rewired network with the lowest Shannon entropy with the community structure minimizing the cost.

The cost of uncertainty, κ, is a function comparing two communities. In the cost minimizing community structure construction one argument is the newly constructed k version of the candidate

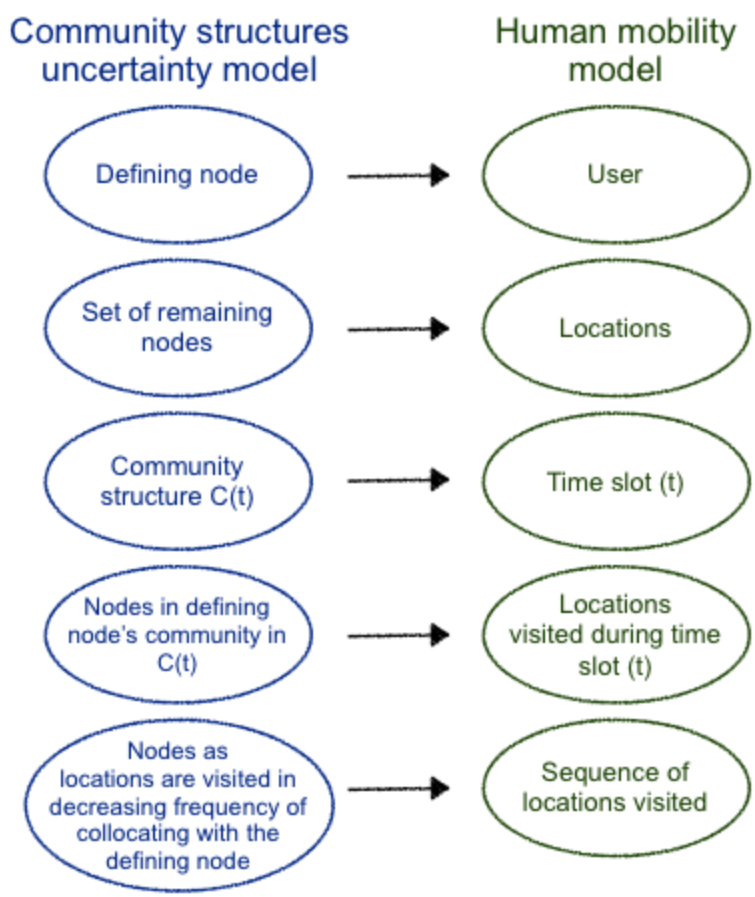

Figure 4.2: Mapping the node's community structure uncertainty model onto the real entropy metric in the human mobility model.

community structure C_k^{can} and one of the s already established structures C_i. Its definition is:

$$\kappa^T(C_k^{can}) = \sum_{i=1}^{s} \kappa(C_k^{can}, C_i), \tag{4.2}$$

where κ^T denotes the total cost of uncertainty for a community structure against all its rewired copies.

We define two different cost functions $\kappa(C_k^{can}, C_i)$. The simple cost function, termed a *frequency-based*, accounts for the average frequency of pairs of nodes appearing in all ground truth communities. It is defined as follows:

$$\kappa_{freq}(C_k^{can}, C_i) = \sum_{j=1}^{|N|} \left(|c_{k,j}^{can} \cup c_{i,j}| - |c_{k,j}^{can} \cap c_{i,j}| \right) f_i^C. \tag{4.3}$$

where $c_{k,j}^{can}$ denotes the community with node n_j in C_k^{can} while $c_{i,j}$ refers to the community with node n_j in community structure i. Hence, this metric penalizes unmatched members of either community with a unit cost, independently of the community size. We can simplify the formula by denoting a size of two community union as $u(i,j,k) = |c_{k,j}^{can} \cup c_{i,j}|$ and a difference between sizes of a union and intersection of the two communities as $d(i,j,k) = \left(u(i,j,k) - |c_{k,j}^{can} \cap c_{i,j}|\right) f_i^C$, yielding

$$\kappa_{freq}(C_k^{can}, C_i) = \sum_{j=1}^{|N|} d(i,j,k). \tag{4.4}$$

The more subtle cost function, referred to as *fraction-based*, is defined as follows:

$$\kappa_{frac}(C_k^{can}, C_i) = \sum_{j=1}^{|N|} \left(1 - \frac{|c_{k,j}^{can} \cap c_{i,j}|}{|c_{k,j}^{can} \cup c_{i,j}|}\right) f_i^C. \tag{4.5}$$

Hence, this metric computes an arithmetic complement of the Jaccard similarity score [86] between pairs of communities sharing a node in the corresponding community structures C_k^{can}, C_i. Unlike the first one, the larger the communities are, the smaller the cost is for mismatched nodes. Again, using $d(), u()$ functions we can simplify the formula as follows

$$\kappa_{frac}(C_k^{can}, C_i) = \sum_{j=1}^{|N|} \frac{d(i,j,k)}{u(i,j,k)}, \tag{4.6}$$

showing similarity between the two cost functions.

The heuristic for creating C^{opt} starts with the initial candidate configuration $C_{can,1}$ in which each node $n_j \in N$ is a community. Let M_j denote the average number of members of communities containing node n_j in all s ground truth community structures. Then, the total cost for the initial structure $C_{can,1}$ is $-|N| + \sum_{j=1}^{|N|} M_j$ for the frequency-based cost. Denoting $m_{i,j}$ the number of members of a community containing node n_j in C_i community structure, the fraction-based cost can be expressed as $|N| - \sum_{j=1}^{|N|} \sum_{i=1}^{s} \frac{f_i^C}{m_{i,j}}$.

For the frequency-based cost, we first compute the weighted average frequency of all pairs of nodes in all s feasible ground truth community structures, and denote it as $f_p(j_1, j_2)$. Consequently, the change of cost from joining j_1, j_2 into one community is $\Delta\kappa = (1 - 2f_p(j_1, j_2)$, and the cost decreases when $f_p(j_1, j_2) > 1/2$. This argument holds if we apply it to the communities c_a, c_b and consider the frequency of their union $c_a \cup c_b$, motivating us to define our heuristic inductively as follows.

1. Initial step 1. The initial $C_{can,1}$ is the set of $|N|$ communities, each containing a different node.

2. Inductive step k, where $1 < k \leq |N|$. Having community structure with $|N|+2-k$ communities from step $k-1$, we measure change of cost from merging any pair of communities in this community structure. Then, we select the pair of communities i_1^{opt}, i_2^{opt} with the lowest cost change $\Delta\kappa$ of the merge. If $\Delta\kappa \geq 0$, then the current community structure $C_{can,k-1}$ is the best, and we stop. Otherwise, we merge communities i_1^{opt}, i_2^{opt} creating C_k^{can} structure with one less community due to the merge, that is with $|N|+1-k$ communities. Thus, this heuristic stops in at most $|N|$ steps, when only one community is left in the candidate community structure.

The heuristic requires careful implementation to be efficient. We compute values of functions $d(), u()$ for all pairs of communities i, j, for the initial candidate community. Then, in each iterative step only the merged community in the candidate structure needs to be recomputed. Since the sum of sizes of all communities is at most $s|N|$, and for candidate communities it is exactly $|N|$, then each step of the iteration has complexity of $O(s|N|^2)$ and therefore the whole algorithm is of complexity $O(s|N|^3)$. However, we observe that s grows slowly with r as do sizes of communities in covert networks with $|N|$. Thus, we can reasonably estimate that the sum of s community sizes is $O(\ln(r)\ln(|N|))$ yielding the total complexity of heuristic as $O(|N|^2\ln(|N|)\ln(r))$.

We define here just two cost functions $\kappa()$, but there are others. For example, in covert networks, the importance of information about the network members may be dependent on the member n_i position h in the hierarchy, n_i^h. Such a cost may be defined as $\kappa_{hier}(n_i^h) = 2^{h-1}$. This function could be used alone or with a combination with κ_{freq} or κ_{frac} functions and used in aiding decisions about interdiction of criminal organizations.

4.4 Validation

4.4.1 Synthetic Networks

We run validation on two types of synthetic networks, the first is a star network, and the second is a complete network. For each of those networks, we created two size variants, one with 16 nodes and the other with 32 nodes, and three undirected edge variants, the first is unweighted, the second with all edge weights set to 20, and the third with all edge weights set to 100. The goal of using different sizes, and edge weights is to better understand the affect of size and edge weights on

the set of networks rewired from each synthetic network. Using the mentioned synthetic networks will allow us to compare the results obtained to the results that we expect, which will ultimately better our understanding of the rewiring process, and the application of the entropy-based metrics.

For each of the mentioned synthetic networks variants, we rewire $r = 1000$ networks, using the BWRN generator with p_B values = $[0.875, 0.9375, 0.98]$, and use the Louvain community detection algorithm to detect the community structures of the rewired networks. We then calculate the Shannon entropy, and the set entropy metrics, and their corresponding predictability, on the set of rewired networks' community structures, as shown in Tables 4.1 and 4.2. Noted that for the synthetic networks used, we only measure the uncorrelated-temporal entropy set metric, and its corresponding predictability.

Table 4.1: We rewire each variant of the synthetic star network 1000 **times, using the BWRN generator with varying p_B values, and we report the Shannon entropy, and the Π^{unc}, on the set of rewired networks' community structures.**

p_B values	Entropy metric	16-nodes			32-nodes		
		w = 1	w = 20	w = 100	w = 1	w = 20	w = 100
0.875	Shannon's entropy	3.5214	0	0	4.5306	0	0
	Π^{unc}	31.35%	100%	100%	12.14%	100%	100%
0.9375	Shannon's entropy	2.6769	0	0	4.1088	0	0
	Π^{unc}	44.81%	100%	100%	16.95%	100%	100%
0.98	Shannon's entropy	1.4805	0	0	2.4203	0	0
	Π^{unc}	68.37%	100%	100%	51.51%	100%	100%

Table 4.2: We rewire each variant of the synthetic complete network 1000 **times, using the BWRN generator with varying p_B values, and we report the Shannon entropy, and the Π^{unc}, on the set of rewired networks' community structures.**

p_B values	Entropy metric	16-nodes			32-nodes		
		w = 1	w = 20	w = 100	w = 1	w = 20	w = 100
0.875	Shannon's entropy	4.6052	4.5636	4.5522	4.6052	4.6052	4.6052
	Π^{unc}	1.87%	2.97%	3.19%	1%	1%	1.25%
0.9375	Shannon's entropy	4.5952	4.5583	0	4.6052	4.6052	4.5854
	Π^{unc}	1.97%	3.06%	100%	1%	1.09%	2.18%
0.98	Shannon's entropy	4.425	0	0	4.6052	4.5564	0
	Π^{unc}	8.30%	100%	100%	1.09%	3.10%	100%

From the set of experiments conducted on the star and complete synthetic networks, we

draw the following conclusions. First, as expected the uncertainty of a network increases as the the size of the network increases. Second, the more complex a network is, the more important the weights of edges in the network become, as shown for the complete network. Increasing the edge weights from 20 to 100 in the synthetic complete network used for rewiring helped reaching 100% predictability, as shown for the 32-node network using edge weights = 100, rather than edge weights = 20, when using $p_B = 0.98$. Finally, we also find that as p_B increases the rewired networks become more similar to the base network used for rewiring, and to each other, and thus the Shannon entropy of the generated networks' community structure decreases, and the Π_{unc} increases. The affect of increasing p_B from 0.9375 to 0.98 is clearly observed when applied to the rewiring of the complete network.

A third type of synthetic network that is commonly studied is the ring network which consists of a ring of n nodes connected by one edge. For such networks, the modularity based community detection methods such as Louvain, are unable to find the true communities of the network due to the resolution limit problem [59], [87], [88]. Thus, when rewiring a synthetic ring network, we find that the size, edge weights, or p_B values used when rewiring the original network, have no affect on the uncertainty or limit of predictability of the rewired networks' community structures. The quality of our generated networks' community structures is limited by the accuracy of the Louvain community detection algorithm, which as we mentioned performs badly on ring networks.

4.4.2 Real Networks

We also run validation on the Caviar and Ciel criminal networks [42], the Jakarta Bombing terrorist network [89], and the Dolphins social network [90].

The Caviar network represents criminals that smuggled hashish and cocaine in Montreal, Canada. The data was collected between 1994 and 1996. During this time, the law enforcement officers only seized shipments of drugs, without making any arrests until the investigation was completed. The Caviar network is a weighted and directed network, where edges represent wire-tapped telephone calls between members of the network.

The Ciel network is also a drug trafficking criminal organization that smuggled hashish from Jamaica into Montreal, Canada. Its data was collected between May 1996 to June 1997. This network is also weighted and directed, with edges representing wiretapped telephone calls among members of the network. Both the Caviar, and Ciel networks were created from data publicly released from court proceedings.

The Jakarta Bombing terrorist network is an undirected weighted network composed of two snapshots showing the network before and after the 2009 Jakarta bombing in Jakarta, Indonesia. We only focus on the pre-attack snapshot as it shows a network that is denser than the post-attack network.

The Dolphins network is a social network of bottlenose dolphins living off the shore of New Zealand, with data collected between 1995 and 2001. This network is unweighted and undirected, with edges representing sightings of dolphins together.

4.4.3 Results

To validate our proposed entropy-based metrics, and the heuristic for constructing the optimal community structure, we use the BWRN generator with parameters $p_B = 0.875$ and $r = 1000$. We apply it to rewire all networks for which we measure the Shannon entropy and the set entropy, since these measurements are done across the community structures of rewired networks.

Table 4.3: The lowest Shannon entropy of networks rewired from the original Caviar, Ciel, Jakarta, and Dolphins networks for range of parameters $p_B = [0.5, 0.75, 0.875, 0.9375]$. The generated networks are split into 10 subsets of 100 networks each, and the Shannon entropy is computed for the best community in each subset. The mean and standard deviation are reported for each case with different p_B values.

	p_B values	0.5	0.75	0.875	0.9375
Ciel	mean	4.366	4.101	3.386	2.539
	σ	0.128	0.126	0.122	0.071
Caviar	mean	3.640	3.246	2.727	2.169
	σ	0.188	0.153	0.136	0.121
Jakarta	mean	2.043	1.379	0.944	0.621
	σ	0.174	0.148	0.060	0.039
Dolphins	mean	6.908	6.908	6.906	6.884
	σ	0	0	0.003	0.001

4.4.3.1 *Finding the Lowest Shannon Entropy Network*

The rewiring of the Caviar, Ciel, Jakarta, and Dolphins networks using the BWRN generator, results in generated networks with varying edges and community structures. Our goal is to find iteratively the rewired network that results in the lowest Shannon entropy when further rewired. We start by setting the original network as g^{opt} for the first step. Then, in the iterative step, we

take g^{opt} found in the previous step and rewire it r times. From the set of rewired networks, we select the community structure C^{opt} with the highest fraction, f_i^C, of rewired networks sharing it, and mark the set of rewired networks with this community structure as candidates. Using the BWRN generator, we rewire every candidate network r times, and mark the candidate network that results in the set of rewired networks with the lowest Shannon entropy as the new g^{opt}. We repeat this process until the lowest Shannon entropy among community structure of the newly rewired networks stops decreasing. Once this happens, we repeat the process one more time, and if the set of newly rewired networks have a Shannon entropy that is once again decreasing, then we continue rewiring until we once again yield the minimum achieved Shannon entropy, otherwise we stop. What we find is that in most cases the Shannon entropy of the networks rewired from g^{opt}, tends to be lower than that of the networks rewired from the original network.

To validate our methods, we measure the uncertainty present across the community structures of all the rewired networks using the Shannon entropy and the set entropy-based metrics at each step of aforementioned iteration. We find that over the first few rounds of rewiring, the Shannon entropy and the set entropy are constantly decreasing, until they reach a minimum value. Once this happens further rewiring would cause the Shannon entropy and set entropy to increase. Interestingly, further rewiring of the resulting networks with an increased entropy creates networks that bring back previously observed minimum values of the Shannon entropy and set entropy.

The presence of many candidate networks to rewire in the set of networks rewired from the g^{opt} network indicates that g^{opt} has low uncertainty. With the large repetition parameter r, rewiring all candidates can be computationally prohibitive. Fortunately, we found a heuristic that chooses the network with the largest modularity and performs well as shown in Fig. 4.3.

Fig. 4.4, shows the set entropy of the networks generated from the best candidate network g^{opt}, presented in Fig. 4.3. After every step of rewiring, we measure S_i^{rand}, S_i^{unc}, and S_i, and their corresponding limits of predictability Π^{rand}, Π^{unc}, and Π^{max} over the set of rewired networks' community structures. S_i^{unc}, and its corresponding limit of predictability Π^{unc}, are the most important measures for this study, as they truly measure the node's structure uncertainty. The temporal-uncorrelated entropy considers the number of unique communities to which the node belongs among all r rewired networks, and the node's frequency of appearance in such communities. Fig. 4.4, shows that Π^{unc} increases as the corresponding Shannon entropy decreases. The Π^{rand}, and Π^{max} are presented for completeness.

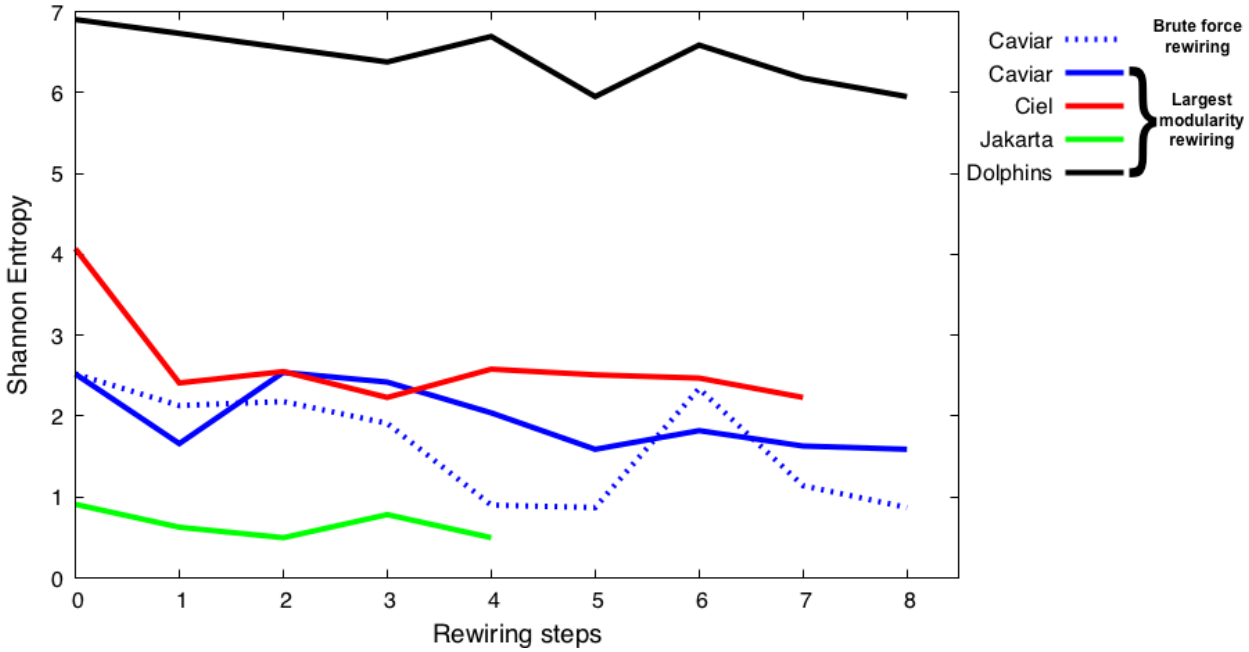

Figure 4.3: A plot of the number of iterations leading to the minimum Shannon entropy for Caviar, Ciel, Jakarta, and Dolphins networks. For the Caviar network we show the results of rewiring using the brute force method, and the heuristic using the largest modularity candidate network. For all the remaining networks, we use the heuristic method.

4.4.3.2 *Convergence of Shannon's Entropy With Extended Rewiring*

We hypothesize that through further rewiring, the Shannon's entropy will eventually converge to the minimum observed Shannon entropy, as reported in Fig. 4.3. To test our hypothesis, we extend the rewiring steps of the Jakarta, and Ciel networks, as they required the minimum number of rewiring steps for the reoccurrence of the minimum Shannon entropy, and thus should converge to the minimum Shannon entropy in the least number of rewiring steps.

In Fig. 4.5 we show the results of the extended rewiring, and confirm our hypothesis. For the Jakarta, and Ciel networks, as we extended the number of rewiring steps, we observed that the Shannon entropy of the rewired networks' community structures starts converging to the minimum Shannon entropy. For more complex networks, such as the Caviar and Jakarta networks, we expect the same trend of convergence to occur, but it may require many more steps of rewiring.

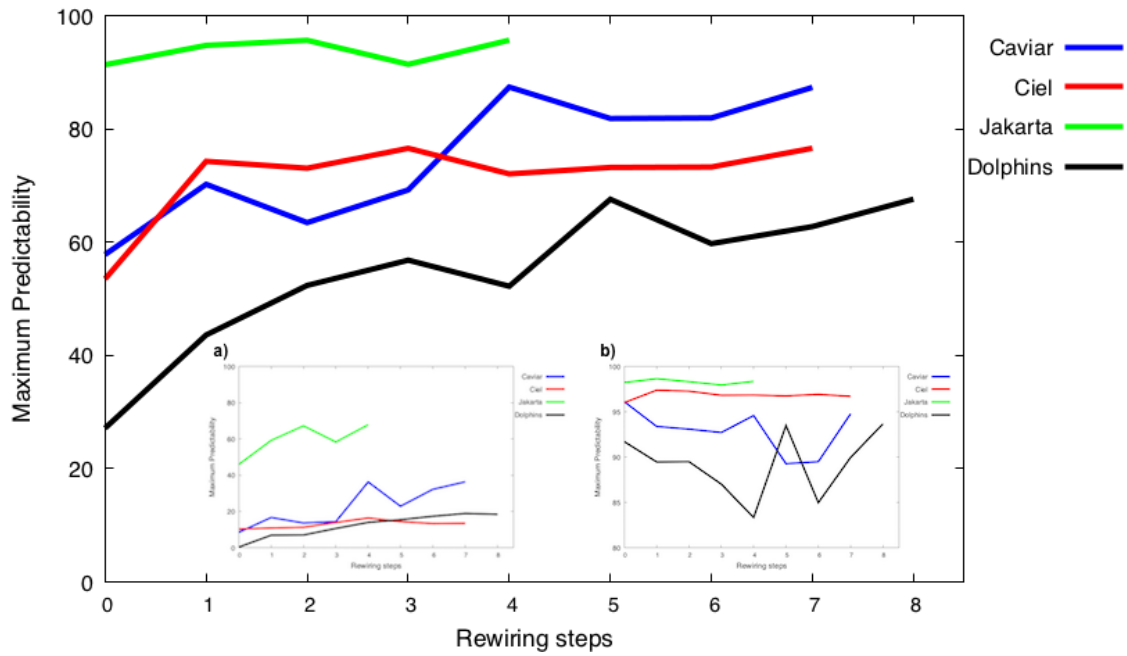

Figure 4.4: During the iterative rewiring of the original Caviar, Ciel, Jakarta, and Dolphins networks presented in Fig. 4.3, we measure the set entropy and the limit of predictability of the set of rewired networks' community structures. The three entropy measures used are S_i^{rand}, S_i^{unc}, and S_i, and we extract from them their corresponding limits of predictability Π^{rand}, Π^{unc}, and Π^{max}. The results for Π^{unc} are presented in the main figure, and the results for Π^{rand} and Π^{max} are presented in insets a and b respectively. We find that as the Shannon entropy of C^{opt} decreases, so does the set entropy of the rewired networks's community structures, and thus the corresponding limit of predictability increases.

4.4.3.3 *Using the Fraction-based and Frequency-based Constructed Communities to Lower Shannon Entropy*

Since the cost-based constructed communities are not limited to those discovered by Louvain algorithm in rewired networks, they can generate a set of rewired networks that have a lower Shannon entropy than the minimum Shannon entropy obtained through the repeated rewiring shown in Fig. 4.3. We found that using the community structure created by the heuristic with fraction-based cost outperforms such a structure created with frequency-based cost in this role. To validate this insight, let R^{opt} stands for the first rewiring step that results in the network g^{opt} yielding the minimum Shannon entropy. We denote C^{frac} the community created by heuristic with κ_{frac} cost function applied to the set of community structures created by the rewiring step R^{opt}.

We find that replacing g^{opt} with the hybrid network with edge structure of g_{opt} and com-

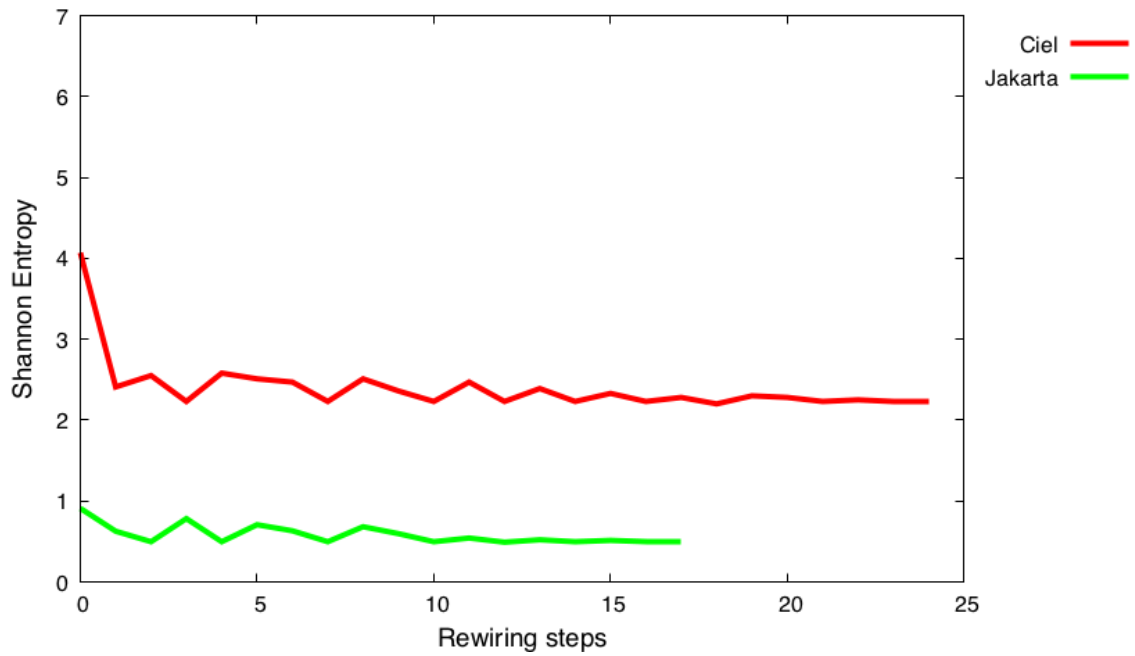

Figure 4.5: We extend the rewiring steps for the Ciel, and Jakarta networks and find that as we hypothesized, the Shannon's entropy of the generated networks' community structures converges as we further extend the rewiring steps.

munity structure C^{frac} for rewiring decreases the Shannon entropy and, accordingly, increases the limits of predictability Π^{unc} and Π^{max}. Table 4.4 shows the results.

To see how minimizing uncertainty and its costs affects the network community and edge structures, we compare these structures to those in the original Ciel, Jakarta, and Dolphins networks. For the three networks, the fractions of edges present in both g^{opt} and in g^{org} ranged from 0.72 to 0.93, while the remaining edges were changed through rewiring. Comparing the community structures C^{frac} and C^{org}, we found that the fractions of nodes maintaining their community memberships ranged from 0.83 to 0.95 and the remaining nodes either created their own new communities, or changed their memberships to already established communities. Comparing the community structures of C^{frac} and C^{opt}, we find that the fractions of the nodes maintaining their community memberships ranged from 0.91 to 0.96, while the remaining nodes created their own new community in C^{opt}. In conclusion, more flexibility in creating communities enables cost-based heuristic to increase changes in its community structure and decrease both the uncertainty and its cost.

Table 4.4: Let g^{opt} denote such network with community structure C^{opt}, which when rewired g^{opt} yields the set of community structures against which g^{opt} achieves the minimum Shannon entropy. Using the BWRN generator, we further rewire a hybrid networks that has g^{opt} edge structure but C^{frac} community structure and find that this networks lowers the minimum Shannon entropy and accordingly increases Π^{unc} and Π^{max} limits of predictability. For the Caviar network, the C^{frac}, and C^{opt} had identical community structures, and thus the Caviar results are omitted here. For comparison, we also measure the uncertainty of the set of networks rewired using the original network, and its corresponding community structure, denoted as C^{org}. Finally, we report the fraction-based cost for each of the mentioned community structures relative to their sets of rewired networks' community structures.

Network	Community structure	Frac cost	Freq cost	Shannon entropy	Π^{unc}
Ciel	C^{org}	659.23	9964	4.07	53.49%
	C^{opt}	231.34	2072	2.23	76.58%
	C^{frac}	206.13	1886	2.12	79.63%
Jakarta	C^{org}	100.19	944	0.91	91.32%
	C^{opt}	25.0	242	0.50	95.64%
	C^{frac}	22.7	227	0.42	96.18%
Dolphins	C^{org}	4573.84	57448	6.90	27.16%
	C^{opt}	1498.09	23864	5.95	67.56%
	C^{frac}	1027.95	15954	5.79	68.12%

4.5 Conclusions

The main contributions of this paper are as follows.

We introduce entropy measures of network community structure and use them to establish limits of such structure's uncertainty. This enables us to search for the network g^{opt} with community structure with the lowest uncertainty against the network structures rewired from g^{opt}. We find such network by iteratively rewiring a network with the smallest Shannon entropy among all rewired networks. We stop this iteration when the current step returns higher uncertainty than the previous one did.

Despite the high theoretical value of low uncertainty, in applications, even more important is its costs, as it can vary widely depending on which nodes are assigned to different communities than predicted. To address this challenge, we introduce a novel heuristic that constructs a community structure with the minimal expected cost from uncertainty of community memberships. The cost function can be predefined or provided by the users based on the application. We show three

examples of such functions.

Finally, we develop a methodology that starts with noisy network data and constructs communities and edges that minimize the expected cost arising from data uncertainty.

In future work, we plan to extend the methodology to other domains, in which networks are created from noise data.

CHAPTER 5
CONTRIBUTION

In Chapter 2, we study the crime rates in 77 community areas in the city of Chicago, with data collected over 16 consecutive years. We develop a predictive algorithm that makes use of both the census and crime data collected for each community area. Through the usage of the null hypothesis test to identify the features that are of importance for the predictive algorithm, we find that the most significant features for the predictive algorithm are crime related features, and that the census data was not of much importance. We also run a crime reduction simulation that shows us the impact of deploying additional crime reduction funding to the community areas, and how such deployment can impact the overall city wide crime rates. What we found is that the deployment of such crime reduction funding in different community areas may result in different city wide crime reduction. Thus, we conclude that some community areas are more influential than others, and that before deploying any crime reduction funding each community area must be carefully analyzed to successfully choose the most influential community area to deploy such funding.

In Chapter 3, we first study the application of social search on a real world network, and study the manipulation of such network can impact the accuracy and efficiency of the social search. Then, we describe the design of a synthetic network generator that creates synthetic networks that are statistically similar to a real-world network. Starting with the social search, we study the Gowalla social network, and we emulate Milgram's social search experiment on this network. We then manipulated the network in several different ways to see how the social search will be impacted. First, we tried different node distributions, by reassigning the location of each user in the network, while maintaining the friendship edges of each user. We found that node displacements had almost no impact on the accuracy and efficiency of the social search. Then we tried different friendship distributions and we found that the actual friendships do not matter, as long as the friendship distribution preserves the degree, and range of the friendships present in the original network. Finally, we introduce the notion of friends of friends, to which we refer to as i-friends. We allow each user to know a maximum of κ (which defines the maximum number of i-friends that a node can know) i-friends for each of their direct friends, and we found that the knowledge of i-friends greatly improves the social search efficiency and accuracy. We conclude that certain friendships distributions will break the properties of the network, and thus negatively impact the social search, and that the knowledge of i-friends drastically improves the social search efficiency

and success rate. In the second part of Chapter 3, we introduce a Bernoulli Weighted Random Network (BWRN) generator, that uses two models for creating synthetic networks. The BWRN uses the Stochastic Block Model for preserving the community structure of the network, and a hierarchical model for preserving the network's hierarchy. The generator models weighted edges in the synthetic networks using a set of Bernoulli trials. We conclude that accounting for both the community structure and the management hierarchy is essential for generating synthetic networks that are statistically similar to the original real network.

In Chapter 4 we use the synthetic networks generated from a real network using the BWRN generator. We introduce entropy-based metrics to assess the probability that a given community structure is the ground truth community structure. Using the Shannon entropy and set entropy-based metrics, we were able to identify the network with the lowest community structure uncertainty. We also introduced two heuristics for creating community structures, using the entire set of BWRN generated networks' community structures, and found that using such community structures in place of the ground truth community structures, results in a set of rewired networks with decreasing uncertainty, and an increasing level of predictability.

REFERENCES

[1] X. Niu, A. Elsisy, N. Derzsy, and B. K. Szymanski, "Dynamics of crime activities in the network of city community areas," *Appl. Netw. Sci.*, vol. 4, no. 1, p. 127, Dec. 2019.

[2] B. A. Furtado, P. A. Sakowski, and M. H. Tóvolli, "Modeling complex systems for public policies," Jul. 2015.

[3] E. Griffiths and J. M. Chavez, "Communities, street guns and homicide trajectories in chicago, 1980–1995: merging methods for examining homicide trends across space and time," *Criminology*, vol. 42, no. 4, pp. 941–978, Mar. 2006.

[4] S. Guarino-Ghezzi and A. J. Treviño, *Understanding Crime: A Multidisciplinary Approach*. Oxfordshire, U.K.: Routledge, 2010.

[5] M. R. D'Orsogna and M. Perc, "Statistical physics of crime: A review," *Phys. life Rev.*, vol. 12, no. 1, pp. 1–21, Mar. 2015.

[6] L. Anselin, J. Cohen, D. Cook, W. Gorr, and G. Tita, "Spatial analyses of crime," *Criminal Justice*, vol. 4, no. 2, pp. 213–262, Mar. 2000.

[7] L. Backstrom, E. Sun, and C. Marlow, "Find me if you can: improving geographical prediction with social and spatial proximity," in *Proc. 19th Int. Conf. World wide web*, 2010, pp. 61–70.

[8] M. B. Gordon, "A random walk in the literature on criminality: A partial and critical view on some statistical analyses and modelling approaches," *Eur. J. Appl. Math.*, vol. 21, no. 4-5, pp. 283–306, Apr. 2010.

[9] A. T. Murray and T. H. Grubesic, "Exploring spatial patterns of crime using non-hierarchical cluster analysis," in *Crime Model and Mapping Using Geospatial Technologies*. New York, NY, USA: Springer, 2013, pp. 105–124.

[10] A. M. Zeoli, J. M. Pizarro, S. C. Grady, and C. Melde, "Homicide as infectious disease: Using public health methods to investigate the diffusion of homicide," *Justice Quart.*, vol. 31, no. 3, pp. 609–632, Oct. 2012.

[11] M. Misyrlis, C. M. Cheung, A. Srivastava, R. Kannan, and V. Prasanna, "Spatio-temporal modeling of criminal activity," in *Proc. 2nd Int. Workshop Soc. Sensing*, 2017, pp. 3–8.

[12] E. K. Weisburst, "Safety in police numbers: Evidence of police effectiveness from federal cops grant applications," *Amer. Law and Econ. Rev.*, vol. 21, no. 1, pp. 81–109, Nov. 2018.

[13] S. Mello, "More cops, less crime," *J. Public Econ.*, vol. 172, no. 1, pp. 174–200, Apr. 2019.

[14] A. Elsisy, B. K. Szymanski, J. A. Plum, M. Qi, and A. Pentland, "A partial knowledge of friends of friends speeds social search," *PLoS One*, vol. 16, no. 8, Aug. 2021, art. no. e0255982.

[15] A. Elsisy, A. Mandviwalla, B. K. Szymanski, and T. Sharkey, “A network generator for covert network structures,” *Inf. Sci.*, vol. 584, no. 1, pp. 387–398, Nov. 2021.

[16] S. Milgram, “The small world problem.” *Psychol. Today*, vol. 2, no. 1, pp. 60–67, May. 1967.

[17] J. Travers and S. Milgram, “An experimental study of the small world problem.” *Sociometry*, vol. 32, no. 4, pp. 425–443, Dec. 1969.

[18] C. Korte and S. Milgram, “Acquaintance networks between racial groups: Application of the small world method.” *J. Personality and Soc. Psychol.*, vol. 15, no. 2, p. 101, Jun. 1970.

[19] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna, “Four degrees of separation.” in *Proc. 4th Annu. ACM Web Sci. Conf.*, 2012, pp. 33–42.

[20] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, “Analysis of topological characteristics of huge online social networking services.” in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 835–844.

[21] H. Kwak, C. Lee, H. Park, and S. Moon, “What is twitter, a social network or a news media?” in *Proc. 19th Int. Conf. World wide Web*, 2010, pp. 591–600.

[22] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, “Measurement and analysis of online social networks.” in *Proc. 7th ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 29–42.

[23] T. Nguyen, M. Chen, and B. K. Szymanski, “Analyzing the proximity and interactions of friends in communities in Gowalla.” in *2013 IEEE 13th Int. Conf. Data Mining Workshops*, 2013, pp. 1036–1044.

[24] D. Laniado, Y. Volkovich, S. Scellato, C. Mascolo, and A. Kaltenbrunner, “The impact of geographic distance on online social interactions,” *Inf. Systems Frontiers*, vol. 20, no. 6, pp. 1203–1218, Aug. 2018.

[25] P. Erdös and A. Rényi, “On random graphs. I,” *Publ. Math.*, vol. 6, no. 4, pp. 290–297, Dec. 1959.

[26] B. L. Welch, “The generalization of student’s’ problem when several different population variances are involved,” *Biometrika*, vol. 34, no. 1/2, pp. 28–35, Jan. 1947.

[27] T. Nguyen and B. K. Szymanski, “Using location-based social networks to validate human mobility and relationships models.” in *Adv. in Soc. Netw. Anal. and Mining (ASONAM), 2012 IEEE/ACM Int. Conf.*, 2012, pp. 1215–1221.

[28] J. Xie and B. K. Szymanski, “Labelrank: A stabilized label propagation algorithm for community detection in networks,” in *2013 IEEE 2nd Netw. Sci. Workshop (NSW)*, 2013, pp. 138–143.

[29] A.-L. Barabási, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: the topology of the world-wide web," *Physica A: Statist. Mech. and its Appl.*, vol. 281, no. 1-4, pp. 69–77, Jun. 2000.

[30] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Sci.*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

[31] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Soc. Netw.*, vol. 5, no. 2, pp. 109–137, Jun. 1983.

[32] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, no. 1, Jan. 2011, art. no. 016107.

[33] M. E. Newman, "Equivalence between modularity optimization and maximum likelihood methods for community detection," *Phys. Rev. E*, vol. 94, no. 6, Nov. 2016, art. no. 052315.

[34] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, no. 4, Oct. 2008, art. no. 046110.

[35] J. Xiao, H.-F. Ren, and X.-K. Xu, "Constructing real-life benchmarks for community detection by rewiring edges," *Complexity*, vol. 2020, Apr. 2020.

[36] E. Ravasz and A. L. Barabási, "Hierarchical organization in complex networks," *Phys. Rev. E*, vol. 67, no. 2, Feb. 2003, art. no. 026112.

[37] K. He, Y. Li, S. Soundarajan, and J. E. Hopcroft, "Hidden community detection in social networks," *Inf. Sci.*, vol. 425, pp. 92–106, Jan. 2018.

[38] K.-k. Shang, B. Yang, J. M. Moore, Q. Ji, and M. Small, "Growing networks with communities: A distributive link model," *Chaos*, vol. 30, no. 4, Apr. 2020, art. no. 041101.

[39] W. Guo and S. B. Kraines, "A random network generator with finely tunable clustering coefficient for small-world social networks," in *2009 Int. Conf. Comput. Aspects Soc. Netw.*, 2009, pp. 10–17.

[40] X. Lu, E. Bulut, and B. Szymanski, "Towards limited scale-free topology with dynamic peer participation," *Comput. Netw.*, vol. 106, no. 1, pp. 109–121, Sep. 2016.

[41] O. Benyahia, C. Largeron, B. Jeudy, and O. R. Zaïane, "Dancer: Dynamic attributed network with community structure generator," in *Joint Eur. Conf. Mach. Learn. and Knowl. Discovery in Databases*, 2016, pp. 41–44.

[42] C. Morselli, *Inside Criminal Networks*. vol. 8, New York, NY, USA: Springer, 2009.

[43] A. Bahulkar, B. K. Szymanski, N. O. Baycik, and T. C. Sharkey, "Community detection with edge augmentation in criminal networks," in *2018 IEEE/ACM Int. Conf. Adv. in Soc. Netw. Anal. and Mining (ASONAM)*, 2018, pp. 1168–1175.

[44] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *J. Stat. Mech.: Theory Exp.*, vol. 2008, no. 10, Oct. 2008, art. no. P10008.

[45] S. K. R. Unnithan, B. Kannan, and M. Jathavedan, “Betweenness centrality in some classes of graphs,” *Int. J. Combinatorics*, vol. 2014, no. 1, Mar. 2014, art. no. 241723.

[46] G. Berlusconi, “Do all the pieces matter? assessing the reliability of law enforcement data sources for the network analysis of wire taps,” *Glob. Crime*, vol. 14, no. 1, pp. 61–81, Dec. 2013.

[47] D. B. Skillicorn, Q. Zheng, and C. Morselli, “Spectral embedding for dynamic social networks,” in *Proc. 2013 IEEE/ACM Int. Conf. Adv. in Soc. Netw. Anal. and Mining*, 2013, pp. 316–323.

[48] Y.-B. Zhou, S.-M. Cai, W.-X. Wang, and P.-L. Zhou, “Age-based model for weighted network with general assortative mixing,” *Physica A: Statist. Mech. and its Appl.*, vol. 388, no. 6, pp. 999–1006, Mar. 2009.

[49] G. Wen, Z. Duan, G. Chen, and X. Geng, “A weighted local-world evolving network model with aging nodes,” *Physica A: Statist. Mech. and its Appl.*, vol. 390, no. 21-22, pp. 4012–4026, Oct. 2011.

[50] B. Hu, X.-Y. Jiang, J.-F. Ding, Y.-B. Xie, and B.-H. Wang, “A weighted network model for interpersonal relationship evolution,” *Physica A: Statist. Mech. and its Appl.*, vol. 353, no. 1, pp. 576–594, Aug. 2005.

[51] D. Shanafelt, K. Salau, and J. Baggio, “Do-it-yourself networks: a novel method of generating weighted networks,” *Royal Soc. Open Sci.*, vol. 4, no. 11, Nov. 2017.

[52] P. Li, J. Yu, J. Liu, D. Zhou, and B. Cao, “Generating weighted social networks using multigraph,” *Physica A: Statist. Mech. and its Appl.*, vol. 539, Feb. 2020, art. no. 122894.

[53] M. C. Davis, Z. Ma, W. Liu, P. Miller, R. Hunter, and F. Kee, “Generating realistic labelled, weighted random graphs,” *Algorithms*, vol. 8, no. 4, pp. 1143–1174, Dec. 2015.

[54] S. Fortunato, “Community detection in graphs,” *Phys. Rep.*, vol. 486, no. 3, pp. 75–174, Feb. 2010.

[55] D. Garlaschelli, “The weighted random graph model,” *New J. Phys.*, vol. 11, no. 7, Jul. 2009, art. no. 073005.

[56] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. New York, NY, USA: Willey, 1968.

[57] C. Gaiteri, M. Chen, B. Szymanski, K. Kuzmin, J. Xie, C. Lee, T. Blanche, E. Chaibub Neto, S.-C. Huang, T. Grabowski, T. Madhyastha, and V. Komashko, “Identifying robust communities and multi-community nodes by combining top-down and bottom-up approaches to clustering,” *Sci. Rep.*, vol. 5, no. 1, Nov. 2015.

[58] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New J. Phys.*, vol. 11, no. 3, Mar. 2009, art. no. 033015.

[59] V. Traag, P. Van Dooren, and Y. Nesterov, "Narrow scope for resolution-limit-free community detection," *Phys. Rev. E*, vol. 84, no. 1, Jul. 2011, art. no. 016114.

[60] M. Chen, K. Kuzmin, and B. Szymanski, "Community detection via maximization of modularity and its variants," *IEEE Trans. Comput. Soc. Sys.*, vol. 1, no. 1, pp. 46–65, Mar. 2014.

[61] X. Lu, K. Kuzmin, M. Chen, and S. B.K., "Adaptive modularity maximization via edge weighting scheme," *Inf. Sci.*, vol. 424, no. 1, pp. 55–68, Jan. 2018.

[62] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, Dec. 2004, art. no. 066111.

[63] S. Rajeh, M. Savonnet, E. Leclercq, and H. Cherifi, "Interplay between hierarchy and centrality in complex networks," *IEEE Access*, vol. 8, no. 1, pp. 129 717–129 742, Jul. 2020.

[64] B. Davison, "Management span of control: how wide is too wide?" *J. Bus. Strategy*, Aug. 2003.

[65] T. E. Oliphant, *A guide to NumPy*. vol. 1, New York, NY, USA: Trelgol Publishing, 2006.

[66] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Stat. Mech.: Theory Exp.*, vol. 2005, no. 9, Sep. 2005, art. no. P09008.

[67] M. J. Zaki and W. J. Meira, *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2020.

[68] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Assoc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, Mar. 2007.

[69] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins, "Microscopic evolution of social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. discovery and data mining*, 2008, pp. 462–470.

[70] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Analyzing communities and their evolutions in dynamic social networks," *ACM Trans. Knowl. Discovery From Data (TKDD)*, vol. 3, no. 2, pp. 1–31, Apr. 2009.

[71] M. Coscia and F. M. Neffke, "Network backboning with noisy data," in *2017 IEEE 33rd Int. Conf. on Data Eng. (ICDE)*, 2017, pp. 425–436.

[72] R. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N. J. Krogan, S. Chung, A. Emili, M. Snyder, J. F. Greenblatt, and M. Gerstein, "A bayesian networks approach for predicting protein-protein interactions from genomic data," *Sci.*, vol. 302, no. 5644, pp. 449–453, Oct. 2003.

[73] F. Calderoni, D. Brunetto, and C. Piccardi, "Communities in criminal networks: A case study," *Soc. Netw.*, vol. 48, no. 1, pp. 116–125, Jan. 2017.

[74] L. Cavallaro, A. Ficara, P. De Meo, G. Fiumara, S. Catanese, O. Bagdasar, W. Song, and A. Liotta, "Disrupting resilient criminal networks through data analysis: The case of sicilian mafia," *PLoS One*, vol. 15, no. 8, Aug. 2020, art. no. e0236476.

[75] L. Skrabanek, H. K. Saini, G. D. Bader, and A. J. Enright, "Computational prediction of protein–protein interactions," *Mol. Biotechnology*, vol. 38, no. 1, pp. 1–17, Aug. 2008.

[76] N. Bhardwaj and H. Lu, "Correlation between gene expression profiles and protein–protein interactions within and across genomes," *Bioinf.*, vol. 21, no. 11, pp. 2730–2738, Mar. 2005.

[77] H. Cherifi, G. Palla, B. K. Szymanski, and X. Lu, "On community structure in complex networks: challenges and opportunities," *Appl. Netw. Sci.*, vol. 4, no. 117, pp. 1–35, Dec. 2019.

[78] B. Karrer, E. Levina, and M. E. Newman, "Robustness of community structure in networks," *Phys. Rev. E*, vol. 77, no. 4, Apr. 2008, art. no. 046119.

[79] L. Yin and Y. Deng, "Toward uncertainty of weighted networks: An entropy-based model," *Physica A: Statist. Mech. and its Appl.*, vol. 508, no. 1, pp. 176–186, Oct. 2018.

[80] T. Wen and Y. Deng, "The vulnerability of communities in complex networks: An entropy approach," *Rel. Eng. & Syst. Saf.*, vol. 196, no. 1, Apr. 2020, art. no. 106782.

[81] Q. Zhang, M. Li, and Y. Deng, "Measure the structure similarity of nodes in complex networks based on relative entropy," *Physica A: Statist. Mech. and its Appl.*, vol. 491, no. 1, pp. 749–763, Feb. 2018.

[82] M. Kulisiewicz, P. Kazienko, B. K. Szymanski, and R. Michalski, "Entropy measures of human communication dynamics," *Sci. Rep.*, vol. 8, no. 1, pp. 1–8, Oct. 2018.

[83] S. Legramanti, T. Rigon, D. Durante, and D. B. Dunson, "Extended stochastic block models with application to criminal networks," *arXiv preprint arXiv:2007.08569*, Jan. 2020.

[84] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Sci.*, vol. 327, no. 5968, pp. 1018–1021, Feb. 2010.

[85] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Comput. and Commun. Rev.*, vol. 5, no. 1, pp. 3–55, Jul. 2001.

[86] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb. 1912.

[87] A. Khadivi, A. A. Rad, and M. Hasler, "Network community-detection enhancement by proper weighting," *Phys. Rev. E*, vol. 83, no. 4, Apr. 2011, art. no. 046104.

[88] B. H. Good, Y.-A. De Montjoye, and A. Clauset, "Performance of modularity maximization in practical contexts," *Phys. Rev. E*, vol. 81, no. 4, Apr. 2010, art. no. 046106.

[89] S. F. Everton, *Disrupting Dark Networks*. no. 34, Cambridge, U.K.: Cambridge Univ. Press, 2012.

[90] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behav. Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, Jun. 2003.