Locating People of Interest in Social Networks

Pivithuru Wijegunawardana
Syracuse University

ABSTRACT

By representing relationships between social entities as a network, researchers can analyze them using a variety of powerful techniques. One key problem in social network analysis literature is identifying certain individuals (key players, most influential nodes) in a network. We consider the same problem in this dissertation, with the constraint that the individuals we are interested in identifying (People of Interest) are not necessarily the most important nodes in terms of the network structure. We propose an algorithm to *find POIs*, algorithms to *collect data to find POIs*, a framework to *model POI behavior* and an algorithm to *predict POIs* with guaranteed error rates.

First, we propose a multi-objective optimization algorithm to find individuals who are expected to become stars in the future (rising stars), considering dynamic network data and multiple data types. Our algorithm outperforms the state of the art algorithm to find rising stars in academic data.

Second, we propose two algorithms to collect data in a network crawling setting to locate POIs in dark networks. We consider potential errors that adversarial POIs can introduce to data collection process to hinder the analysis. We test and present our results on several real-world networks, and show that the proposed algorithms achieve up to a 340% improvement over the next best strategy.

Next, We introduce the Adversarial Social Network Analysis game framework to model adversarial behavior of POIs towards a data collector in social networks. We run behavior experiments in Amazon Mechanical Turk and demonstrate the validity of the framework to study adversarial behavior by showing, 1) Participants understand their role, 2) Participants understand their objective in a game and, 3) Participants act as members of the adversarial group.

Last, we show that node classification algorithms can be used to predict POIs in social networks. We then demonstrate how to utilize conformal prediction framework [103] to obtain guaranteed error bounds in POI prediction. Experimental results show that the Conformal Prediction framework can provide up to a 30% improvement in node classification algorithm accuracy while maintaining guaranteed error bounds on predictions.

LOCATING PEOPLE OF INTEREST IN SOCIAL NETWORKS

By

Pivithuru Wijegunawardana
B.Sc (Hons.) in Computer Science and Engineering, University of Moratuwa, 2011
M.S. in Computer Science, Syracuse University, 2019

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer & Information Science & Engineering

Syracuse University
December 2019

Acknowledgments

There are many people who supported me throughout my PhD. First and foremost, I am grateful to my advisor Prof. Sucheta Soundarajan for the guidance she provided throughout my PhD. She not only taught me how to think like a researcher but also encouraged me to explore problems that have a greater benefit to the society. I would also like to thank Prof. Soundarajan for countless hours she spent discussing research with me and editing my manuscripts. Prof. Soundarajan is the most supportive advisor anyone can ask for. She was a wonderful role model to me throughout my PhD.

I would also like to thank Prof. Kishan Mehrotra and Prof. Chilukuri Mohan for being my advisors at the beginning of my PhD. They helped me understand and navigate research as a new PhD student. I am also thankful to my Phd committee members, Prof. Jae Oh, Prof. Ramesh Raina, Prof. Chilukuri Mohan, Prof. Qinru Qiu and Prof. Fanxin Kong for spending their valuable time reading my dissertation and attending my defense.

My Phd research was supported by the U.S. Army Research Office under grant number W911NF1810047. I would like to extend my thanks to my research collaborator, Dr. Ralucca Gera for spending her time discussing research with me and providing me valuable feedback.

I am also thankful to my past lab mates at SENSE lab and current lab mates at SUNS lab for all research discussions, paper reading groups and the parties we had. Spending time with you made my PhD life joyful. I should specially mention my partners in crime Ricky Laishram, Zhiruo Zhao and Katchaguy Areekijseree for all the wonderfully meaningless conversations we had.

I extend my heartfelt thanks to all my professors at Syracuse University, University of Moratuwa and my teachers from St. Paul's Girls' School, Sri Lanka. I was fortunate enough to learn from such a wonderful group of people. I would also like to thank, Chulaka Gunasekera, Umashanthi Pavalanathan and Sachini Weerawardhana for encouraging me to pursue a carrier in research and helping me in numerous ways throughout my PhD.

Most of all, I am indebted to my parents for believing in me. If not for their unconditional love, support and encouragement I would not be here today. I am also grateful to my sisters for their support. Last but of course not the least, I would like to thank my husband for always being the strength behind me. I thank him for understanding my unusual work schedules, staying up late with me and making me smile through difficult times. I cannot thank him enough for all the sacrifices he made to come to this foreign land to be with me. I dedicate this dissertation to my husband, the love of my life.

Contents

Acknowledgments **v**

List of Tables **x**

List of Figures **xii**

1 Introduction **1**

1.1 Network Centrality Measures 3

1.2 Identifying a Group of People 4

1.2.1 Community Detection 5

1.2.2 Node Classification 6

1.3 Locating People in Social Networks 6

1.4 Our Contributions 7

2 Finding Rising Stars in Social Networks **10**

2.1 Background 11

2.2 Multi-Objective Optimization (MOO) Approach 13

2.3 Methodology for Academic Domain 14

2.3.1 Co-authorship Network 15

2.3.2 Author Citation Network 16

2.3.3 Author Publication Venue Network 17

2.3.4 PageRank Algorithm 19

2.4 Proposed Algorithm 19

2.4.1 Multi-Objective Optimization Approach to Find Rising Stars 20
2.4.2 Rank Aggregation to Find Rising Stars 21
2.5 Experimental Evaluation 22
2.5.1 Multi-Objective Optimization Approach Results 23
2.5.2 Thin Layer of Non-dominated Individuals 25
2.5.3 Rank Aggregation Approach Results 26
2.6 Finding Rising Stars in Q&A Forums 27
2.6.1 Network Formulation 28
Question-Answer Network 28
Question-Comment Network and Answer-Comment Networks . . . 29
2.6.2 Results and Evaluation 29
2.7 Conclusion 30

3 Sampling Social Networks to Locate People of Interest 32
3.1 Problem Definition 35
3.1.1 Problem Setting 1: Node color reliable 36
3.1.2 Problem Setting 2: Node color misreported 37
3.2 Background 37
3.2.1 Criminal Network Analysis 37
3.2.2 Network Sampling Algorithms 38
3.2.3 Node Classification Algorithms 39
3.3 Proposed Method 41
3.3.1 `RedLearn`: A Learning Based Monitor Placement Algorithm 41
3.3.2 REDLEARN RE-SAMPLING (`RedLearnRS`) Algorithm 44
3.4 Datasets 48
3.4.1 Noordin Top Network 48
3.4.2 Pokec Network 49
3.4.3 Facebook100 Networks 49

3.4.4 Synthetic Terrorist Networks 51

3.5 Experimental Setup 53

3.5.1 Baseline Monitor Placement Algorithms 54

3.5.2 Node Color Errors 55

3.5.3 Edge Existence Errors 55

3.5.4 Edge Non-Existence Errors 56

3.5.5 Neighbor Color Errors 56

3.5.6 Experimental Settings 58

3.6 Results and Analysis 60

3.6.1 Node Classification Algorithms 61

3.6.2 Problem Setting 1: Node color reliable 62

3.6.3 Problem Setting 2: Node colors misreported 64

3.6.4 Edge non-existence error analysis 70

3.6.5 Feature Importance Analysis 71

3.6.6 Network structure analysis 73

3.7 Discussion and Conclusion 76

4 Modeling People of Interest in Social Networks 81

4.1 Background 83

4.2 Adversarial Social Network Analysis Game 84

4.3 Experiments and Results 89

4.3.1 Participants pay attention to key elements of the ASNA game 89

4.3.2 Participants understand objectives of the game 91

4.3.3 Participants are loyal to the Red group 93

4.4 Discussion and Conclusion 95

5 Predicting People of Interest with Bounded Error Rates 97

5.1 Background 99

5.1.1 Node Classification Algorithms 99
5.1.2 Conformal Prediction Framework 100
5.1.3 Related Work 102
5.2 Methodology 104
5.3 Experiments 106
5.3.1 Performance Metrics 106
5.3.2 Datasets 107
5.3.3 Experimental Setup 107
5.4 Results 108
5.4.1 Perturbation Analysis 113
5.4.2 Running Time Analysis 113
5.5 Discussion and Conclusion 113

6 Conclusion 115

Bibliography 118

List of Tables

2.1 Co-author and paper citation network data 16
2.2 Rising stars discovered using Multi-Objective Optimization approach 23
2.3 Profiles of top ten rising stars obtained by using the rank aggregation approach 27
2.4 Q&A forum rising stars results compared with a normal user 30
2.5 Rising stars found using MOO approach for Q&A forum data 30

3.1 Classification features used by `RedLearn` for a node v with neighbor set $N(v)$. Here, c_v is the true color of v, s_v is the suspected color of v and e_{uv} is u's estimate of v's color. 42

3.2 Additional features used by `RedLearnRS` for a node v with neighbor set $N(v)$. Here, s_u represents the suspected color of u, and e_{vu} represents the node color that a monitor on node v reports u to be. 46

3.3 Summary of datasets and node attributes we have used to assign node colors. 50

3.4 Noordin Top network hierarchy assignment 60

4.1 Possible rewards and penalties participants can earn from the data collector and the adversarial group. 87

4.2 Percentage of participants who passed each filtering game. 90

4.3 A table beside a figure 92

5.1 Network dataset statistics 108

5.2 Conformal prediction framework performance using ICA as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run. 110

5.3 Conformal prediction framework performance using Label Propagation as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run. 110

5.4 Conformal prediction framework performance using GCN as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run. 111

5.5 Conformal prediction framework performance using DeepWalk as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run. 111

List of Figures

2.1 Co-authorship and author citation network examples 17

2.2 Publication venue network 18

2.3 Average citations in each year for all MOO approach rising stars with Google Scholar profiles 24

2.4 Comparison of MOO approach and co-author network method proposed by Li et al. [61]; Average H-index of each rising star, Average number of Papers of each rising star, and Average number of citations for each rising star respectively 25

2.5 Performance of MOO approach compared to using co-author PageRank slope, citation PageRank slope, venue score slope separately 25

2.6 Performance of first three layers of Non-dominated rising stars 26

2.7 Stack Exchange Q&A forum heterogeneous graph structure 28

3.1 `RedLearnRS` algorithm flow. This is a re-sampling based algorithm to identify nodes of interest in a network where node colors reported by monitors may contain errors. 47

3.2 Criminal network typologies described by [57]. (a) Standard hierarchy with a chain of commands (b) Regional hierarchy with multiple chains of commands and a central controlling body (c) Clustered hierarchy with tight knit groups (d) Core group hierarchy, which doesn't exhibit clear structure . . . 52

3.3 Comparison of average performance of ICA and RedLearn on the NoordinTop terrorist network and Facebook-100 networks. RedLearn outperforms ICA in all networks for both neighbor color error types considered. 61

3.4 Comparison of monitor placement algorithms on the NoordinComs4 network when reported node colors are reliable (Problem Setting 1). MostRedNeighbors (MRN) performs well when there are no edge existence errors. RedLearn performs consistently well across all edge existence error scenarios. 63

3.5 Monitor placement algorithm average performance when reported node color is reliable (Problem Setting 1). Neighbor color error type 1 (NCE1:All nodes aware of red nodes) is considered here. MostRedNeighbors (MRN) and RedLearn algorithms perform similarly when there are no edge existence errors. As we introduce these errors,RedLearn becomes the clear winner across all monitor placement algorithms. 65

3.6 Monitor placement algorithm average performance when reported node colors are reliable (Problem Setting 1). Neighbor color error type 2 (NCE2: Only red nodes aware of other red nodes) is considered here. RedLearn has matched the performance of MostRedNeighbors (MRN), in most networks when there are no edge existence errors. As red nodes hide their connctions with other red nodes, RedLearn becomes the clear winner. 66

3.7 Comparison of monitor placement algorithms on the NoordinComs4 network when monitors misreport node colors (Problem Setting 2). Even though MostRedNeighbors (MRN) performs well in the original network, its performance worsens when edge existence errors are introduced. RedLearn performs consistently well across all neighbor color error types and edge existence error scenarios. 67

3.8 Monitor placement algorithm average performance when node colors are misreported by monitors (Problem Setting 2), under neighbor color error type 1 (NCE1: all nodes are aware of red nodes). MostRedNeighbors (MRN) performs well when there are no edge existence errors, but worsens as we introduce such errors. `RedLearn` performs consistently well across all edge existence error scenarios. 68

3.9 Average performance of monitor placement algorithms when node colors are misreported (Problem Setting 2), under neighbor color error type 2 (NCE2: Only red nodes are aware of other red nodes). `RedLearn` performs consistently well across all edge existence error scenarios, when other monitor placement algorithms are affected heavily by concealing red edges. . . . 69

3.10 Average performance of monitor placement algorithms in NoordinTop networks when monitors report incorrect edges. Problem Setting 1: Node colors are reliable is considered here. Performance of monitor placement algorithms is not heavily affected by noisy edges. 70

3.11 Average performance of monitor placement algorithms in NoordinTop networks when monitors report incorrect edge. Problem Setting 2: Node colors are misreported is considered here. Performance of monitor placement algorithms remains quite similar to when there are no noisy edges. 71

3.12 Average feature importance scores of NoordinTop and Facebook100 Amherst-Major networks when reported node colors are reliable (Problem Setting 1). Tight knit groups are the most important feature category in NoordinTop networks, since red nodes tend to be clustered together. The importance of the neighbor feature category increases when we hide red edges. 73

3.13 Average feature importance scores of NoordinTop and Facebook100 Amherst-Major networks when node colors may be misreported by monitors. Repeated monitor feature group has the highest importance in most networks since this feature category determines whether to place a monitor on an already monitored node. 74

3.14 Comparison of monitor placement algorithm performance for different criminal network hierarchies when reported node colors are reliable (Problem Setting 1). `RedLearn` performs the best in all network hierarchies in NCE 1. When red nodes are clustered and/or connected to lot of other red nodes (core group), MRN performs well. 75

3.15 Comparison of monitor placement algorithm performance for different criminal network hierarchies when node colors may be misrported by monitors (Problem Setting 2). `RedLearnRS` outperforms all other monitor placement strategies. 77

4.1 An example game network. `Red` nodes are werewolves and `Blue` nodes are villagers. Dark colored nodes have already been investigated. The numbers by the investigated nodes show the order of investigator past choices. mTurk participant plays as the “YOU" node in the network. Participants need to report colors of the green circled neighbors of the “YOU" node in the network. 85

4.2 User interface the participants would interact with to report their answers about neighbors. The rewards table opens when a participant clicks on a neighbor and shows possible rewards she can earn for each possible answer. Potions left indicates how many other nodes, the data collector can investigate after the participant. 86

4.3 Network a) tests whether the likelihood of adversarial nodes misreporting data changes (based on how participants perceive the likelihood of the data collector discovering the truth) and, networks b) and c) test whether participants show loyalty to the Red group. Table shows the percentage of participants, who have misreported information about their neighbors in the networks shown. Statistically significant differences from a one tailed t-test at 0.05 significance are marked with $*, \dagger, \ddagger$ symbols. Cells with the same symbol represent that the larger value is significantly larger than the smaller value. 92

5.1 Comparison of using conformal prediction framework vs class probability as expected error for Cora citation dataset classification using Iterative Classification Algorithm. Conformal prediction actual error is always no greater than the expected error where as the actual error for class probability does not follow the expected error. 98

5.2 Sampling algorithm exchangeability example. 105

5.3 Accuracy and efficiency for Cora citation data set using ICA as the baseline algorithm when 10%, 30% and 50% of the nodes are labeled. The actual error is always no greater than the specified significance level. 109

5.4 Performance of ICP applied to CORA citation data with 30% of nodes initially labeled. ICA is used as the baseline. Note that $10\%, 30\%$ and 50% of training data mislabeled. ICP maintains the error bounds even when 50% of the training data is mislabeled. But the efficiency decrease as there are more errors. 112

CHAPTER 1
INTRODUCTION

Complex interactions between social entities create social networks, where nodes in the network are social entities such as people, animals, organizations, and connections (edges) are interactions among these entities. Some examples of such social networks are friendship networks, email communication networks, phone contact networks, co-authorship networks etc. Representing social interactions as a network provide an ample opportunity to study and analyze social entities. Such representation helps to identify common patterns of interaction among individuals, dense groups who interact with each other more often than others, influential individuals in among these social entities etc.

Social network analysis has gained attention from researchers in many disciplines such as sociology, psychology, criminology, physics and computer science due to insightful representation a network can provide to social interaction data. Social scientists utilize social network analysis to provide explanations to social phenomenas [14]. For example, Granovetter et al. studied types of connections (ex: weak ties vs strong ties) social entities can have and implications of these connections [40]. Klovdahl et al.[55] use social network analysis to study disease spread. The criminology literature use social network analysis to model criminal network hierarchies [57], find vulnerabilities in criminal organizations and disrupt them [92]. Physicists and computer scientists on the other hand are more interested in the algorithmic aspects of social network analysis. They generally propose algorithms to analyze [39, 13] and generate [104, 52] social networks. Our work dives into the multi-disciplinary aspect of social network analysis proposing algorithmic solutions problems that arise in criminology and sociology literature.

Social network analysis research has gained much attention recently, predominantly due to the rise of online social networks such as Facebook, Twitter, Reddit etc. Previously, social network data collection was a challenging task since it involved people participating in surveys or using tracking devices to collect data about which individuals are in close proximity to each other. Such data collection usually results in smaller social networks due to the lengthy collection process. Online social networks provide the opportunity to analyze much larger social networks and test social social phenomena on more real world scale social networks.

Social network analysis research can be broadly categorized into 1) Network data collection / generation (Network crawling, generative models), 2) Finding nodes / edges of interest (identifying key players, link prediction), 3) Understanding structural properties (social network properties, scale free networks, community detection), and 4) Understanding social phenomena (information spread, disease propagation). This dissertation lies within the first two categories. We propose algorithms to locate People of Interest in social networks in this dissertation. These works align more with finding nodes of interest in social network. We also consider this problem in terms of collecting network data, where we propose algorithms to collect data to maximize POIs in the collected sample.

People of Interest (POI) in social networks are those individuals who have some common property that make them important to identify apart from the others. POI attributes on one hand can be solely based on the position of some node in the network. There are many network centrality measures in social network analysis literature to identify key players/ influential nodes in social networks based on their positions in a social network.

On the other hand, POI attributes may not depend only on the network structure. Some examples are, people who are members of a criminal group, people who have similar political views, people who attend the same university etc. In this problem setting, POIs are not necessarily the most important nodes in the network. Rather, they as a group possess some common trait that distinguish them from the other nodes. Some applications of algorithms to find such POIs are, uncovering covert nodes in networks, advertising campaigns that target nodes with some given attribute (ex: age), finding experts/ future experts in some field, friend recommendation systems (ex: recommend friends who have similar hobbies) etc.

1.1 Network Centrality Measures

Finding high centrality nodes in a network is a widely studied topic in social network analysis literature. In some cases, these high centrality nodes may correspond to POIs. Following are some classic network centrality measures.

- **Degree Centrality:** Nodes that are connected to most other nodes in a network. Highest degree nodes in a network have highest degree centrality.
- **Eigenvector Centrality [12]:** Nodes that are connected to nodes with high eigenvector centrality scores get a higher centrality score themselves. This centrality measure considers that connecting to few important people in a network is more beneficial compared to lots of people with lower importance. Eigenvector centrality is calculated using the eigenvalue decomposition of a social network adjacency matrix.

- **PageRank Centrality [79]:** This centrality measure is a variation of Eigenvector centrality. For some node V, PageRank centrality is calculated using the probability, a random walk that starts from any node in a network landing node V. Katz centality [53] is a similar centrality measure.
- **Betweenness Centrality [33]:** This is measure based on shortest paths in a network. Betweenness centrality ranks nodes that are more in shortest path between other nodes higher in centrality.
- **Closeness Centrality [33]:** This measure is also based on shortest paths in a network. Closeness centrality ranks nodes that are are closest to other nodes in the network higher in centrality
- $K-core$ **Centrality [29]:** K-core centrality ranks a high degree node who is densely connected to other high degree nodes higher in centrality.

Many variations of these centrality measures can be found depending on specific applications. For example, there are many variations of betweenness and closeness centrality to find influential people in information spread [15, 31]. These measures can be used to identify POIs when their POI attributes correlate with the network structure. In general cases, when POI attributes do not only depend on the network structure, these measures fail to identify such POIs.

1.2 Identifying a Group of People

Identifying a group of people with some common attribute is closely related to community detection and node classification problems in social network literature.

1.2.1 Community Detection

A community in a social network is a group of nodes who are densely connected to nodes inside the community compared to nodes outside. Community detection is a widely studied problem since the community structure reveals underlying node groups in a social network. Community detection problem is similar to clustering problem in machine learning but with added complexity of connections among entities.

The modularity maximization [25] algorithm finds communities that maximizes internal connections in a community compared to a null model in a way that maximizes the expected number of connections between a group of nodes. Label propagation based algorithms [41] start with a unique label indicating that each node belong to a community of its own. These labels are propagated in the network and a node at each iteration accepts the majority label among neighbors as its label. Densely connected nodes converge to the same label quickly, indicating they belong to the same community. Network flow based algorithms [3, 87] assign PageRank based weight to each node called "flow". Then the algorithms identify network partitions that maximize intra flow and minimize inter flow. These algorithms only consider network structure to find these groups of nodes and cannot be used effectively to identify POIs, when POIs are not densely connected sub groups in a network.

There are community detection algorithms that consider node attributes along with network connectivity [48, 62]. These algorithms assume that node attributes are known and tend to group nodes with similar attributes together. We propose algorithms to find POIs when we do not know which nodes may be have POI attributes beforehand.

1.2.2 Node Classification

Node classification algorithms are semi-supervised prediction algorithms that predict labels for unlabeled nodes in partially labeled networks. The node classification problem is similar to the classification problem in machine learning; however the classification instances are connected to each other. Node classification algorithms performs better than traditional classification algorithms because they not only consider node features, but also network connections to predict labels for unlabeled nodes. Therefore, node classification algorithms can be used to identify POIs even if POI attributes are not correlated with the network structure. We discuss details of node classification problem in Chapter 3 and Chapter 5.

1.3 Locating People in Social Networks

Locating people in social networks can refer to 1) Finding POIs in given a complete social network, 2) Collecting data to maximize the number of POIs in the sample. Finding POIs in a given social network is similar to key player identification problem or node classification problem. Social network data collection is of two forms, network down-sampling and network crawling. Network down-sampling algorithms are designed to obtain an unbiased, representative sample of a larger social network [59]. Our work aligns more with the second type- network crawling algorithms- in which the algorithm starts with the knowledge of few nodes in the network and collect data by querying nodes. Inspiration for most network crawling algorithms comes from popular web crawling algorithms. Therefore, classic web crawling algorithms such as Breadth First Search (BFS), Depth First Search (DFS), Snowball Sampling [9], Random walk sampling [49] can be used to crawl social network data.

There are network crawlers designed to collect data to maximize some objective. For example, Avrachenkov et al. [5], propose an algorithm to maximize the node coverage in a social network by querying nodes, with the highest observed degree in the collected sample so far. Target oriented network crawling [93, 11], crawls networks to locate a set of specified nodes in a social network similar to locating POIs. Similarly, in Chapter 3, we propose algorithms to collect network data to maximize the number of POIs identified in the sample. However, we consider the setting when queries to obtain network data provide purposeful misinformation.

1.4 Our Contributions

We propose algorithms to identify POIs in social networks, when the POI properties cannot be easily explained by only using the network structure. In Chapter 2, we consider the key player identification problem, but predict future key players in social networks considering their behavior in networks over time. We propose algorithms to crawl dark social networks to identify as many criminals as possible in Chapter 3. Even though we consider various errors POIs in dark networks can introduce to data collection in Chapter 3, we do not have realistic models to explain what motivates an individual to become adversarial towards a data collector. In Chapter 4, we propose a framework to conduct behavior experiments to study interactions between a data collector and an adversarial group. Algorithmic predictions can have adverse effects on individuals specially in problems such as identifying criminals in social network. In Chapter 5, we demonstrate how to predict POIs with guaranteed error bounds using node classification algorithms.

- **Finding Rising Stars in Social Networks [106]:** POIs in this problem are individuals who are expected to become stars in a given discipline in the near future. To predict rising stars (important nodes), we need to identify nodes who are improving their importance over time. Further, we need to consider all possible information channels to find enough evidence to predict a node as a rising star. Therefore, we formulate this problem as a dynamic social network analysis problem with heterogeneous information sources to determine importance of individuals. One application of this problem is finding rising researchers in academic domain, where the information sources are co-authorship data, citation data and publication venue data. We propose an algorithm to combine known centrality measures, dynamic change in node centrality and heterogeneous information to successfully predict rising stars in social networks.

- **Sampling Social Networks to Locate People of Interest [107, 108]:** In dark networks, POIs are covert entities who try to hide from being detected. Finding POIs in such a setting is a challenging task, since POIs act adversarial towards those who try to collect their data. Therefore, algorithms to find POIs in such networks need to be robust against misinformation and should not be entirely dependent on the network structure. We present `RedLearn` and `RedLearnRS`, two algorithms to crawl dark networks with the goal of maximizing the identification of people of interest, given a limited sampling budget.

- **Modeling People of Interest in Social Networks [109]:** Extending the work in Chapter 3, we introduce the Adversarial Social Network Analysis game (`ASNA game`), an experimental framework to study the adversarial behavior of covert groups in social networks. We conduct experiments on Amazon Mechanical Turk using the proposed framework, which models interactions between a data collector and members of an adversarial group as a network game. Participants of the experiments play as adversarial nodes in the game.

Our goal is to understand the factors that motivate an individual to report network or attribute data dishonestly. We demonstrate the validity of the `ASNA game` framework by showing that participants understand and pay attention to key elements of the game, they try to maximize their objective in the game, and they show loyalty to the adversarial group.

- **Predicting People of Interest with Bounded Error Rates [105]:** Node classification algorithms can be used to predict POIs in social networks. In many problems, a user may wish to associate a confidence level with a prediction such that the error in the prediction is guaranteed. We propose adopting the Conformal Prediction framework [103] to obtain guaranteed error bounds in node classification problem. We show how this framework can be applied to 1) Obtain predictions with guaranteed error bounds, and 2) improve the accuracy of the prediction algorithms.

Chapter 2
Finding Rising Stars in Social Networks

First, we consider the key player identification problem in social networks. When networks change with time, it is important for decision-makers to identify nodes that are expected to become most influential in the future (rising stars), not just the current influential nodes. For example, when hiring someone, it would be helpful to know whether they have the potential to be a stellar performer. Here, POIs we identify are rising stars in social networks. We define a rising star as an individual who has a low expert score (ex: low PageRank score) at the start but shows a significantly improving expert score in successive time steps compared to other users, and thus possesses characteristics to become a prominent contributor in future.

Suppose that we find most the influential nodes in a social network using some centrality measure. Can we predict rising stars by only looking at the centrality scores in the current network? Node centrality scores in one social network do not provide us with enough information to predict who would posses high centrality scores in the future. Therefore, we need to identify nodes who show improving centrality scores over a period of time. Another important attribute a rising star should showcase is persistence over time. This confirms that the node will continue to improve their centrality scores in the future to become an influential node. We assume that a person shows persistence in some domain if he shows improving centrality scores across all data sources in the domain considered. This indicates that POI (rising star) properties are not only correlated with structure of one network or one type of data. Therefore, we propose to find rising stars in heterogeneous dynamic social networks.

An example problem is the identification of rising stars among academic researchers. At the beginning of their career they would not have many publications, citations or might not publish in the most prestigious conferences/journals. But as time goes on, if they are rising stars, they start to collaborate with other prominent academics, collect more citations and publish in prestigious conferences. In view of their improving profiles over time we predict them as rising stars considering various types of interactions such as coauthor, citation and publication venue relationships.

We evaluate the proposed algorithms using academic data (Arnetminer)[99] and question answer (Q&A) forum data [50]. The proposed algorithms obtain superior results compared with current approaches to find rising stars in the academic domain. In Q&A forum data, we are able to identify promising future experts.

2.1 Background

Li et al. [61] have proposed an author ranking scheme based on publications to find rising stars in a co-authorship network. They have used a directed weighted network in which the weight assigned to the link between two co-authors is based on number of mutual publications and total publications each author has. Further, each author is given a node-weight considering the prestige of the conferences and journals where they have published. Weighted PageRank algorithm is used to measure the expertise of an author and evolution of a profile is measured using linear regression gradient of expertise scores over time.

Extending the work presented in [61], Daud et al. [22] introduced an author-contribution based weight assignment to edges in the co-author network. They argue that the order in which co-authors appear in a publication is an important attribute to assign link weights between two authors.

Both these methods combine author collaboration information and publication venue information into one measure to indicate expertise and predict rising stars using linear regression gradient over time. This approach voids the opportunity to observe the evolution of a user in different aspects. Further they cannot be generalized to any other domain as expertise measures are different in each. For example, citations is an important attribute when looking at the profile of an academic. But this information cannot be integrated into these approaches proposed by Li et al. [61] and Daud et al. [22].

Tsatsaronis et al.[101] have proposed an author profile classification based method to find rising stars. They divide authors into four different categories: (1) Well established, (2) Rising stars, (3) Authors with stable publication Rate, and (4) Declining authors, based on how an author's profile evolves over time using a clustering algorithm. The feature set for the clustering algorithm is derived from the co-author network and publication venues.

Daud et al. [23] use a classification approach where the trained classifier predicts whether a given author is a rising star. They discuss an extensive set of features to classify an author as a rising star. But one major requirement for a classification task is having a labeled dataset. In order to assign the class label, they have used author's citation information. But citation information alone does not provide a complete representation of an author's profile. This suggests labeling rising stars in a data set itself require knowledge of how to find rising stars.

In summary, all the existing approaches to find rising stars focus only on academic domain and they cannot be generalized into other domains. Even in the academic domain most methods don't consider citation information. Integrating citation information into existing methods requires changing the expert score calculation, which is not an easy task.

By contrast, the algorithm we propose to find rising stars, works well in heterogeneous data domains, where multiple types of data is available to define the same individual. New data sources (for example citations in academic domain) can be easily integrated to the problem by defining an expert scores in that particular data source. Then expert scores from all data sources can be combined using one of the proposed methods in this chapter to derive the final set of rising stars.

Integration and analysis associated with heterogeneous data has been studied for link prediction [58, 96, 116], community detection [18, 100] and ranking nodes using centrality measures [60]. In this chapter, we present a solution to identify rising stars in a heterogeneous environment.

2.2 Multi-Objective Optimization (MOO) Approach

In order to find the rising stars, the training set consists of k heterogeneous data sources modeled as k weighted networks, $N_t^1, N_t^2, \dots, N_t^k$ and for each network, data is collected over $t = 1, \dots, T$ time points. In each network, individuals are represented as nodes, and edges represent different characteristics. Using such data, the goal of our study is to identify rising stars, i.e., individuals who are expected to perform better than the other individuals in the future.

We model the rising star problem as a decision level data fusion problem. First, we find a key player score for each node in each data source, $\{N_t^j : t = 1, \ldots, T; j = 1, \ldots, k\}$. Next, for each individual we calculate an *evolution index*, which shows how their key player score improves over time. An example evolution index is the linear regression gradient. Thus, associated with each node we obtain a k-dimensional vector of evolution indices representing the performance of a node over time. To find rising stars with respect to all networks we need to combine all evolution indices.

We combine these evolution indices using two methods. In the first method, we model this as a multi-objective optimization problem [27], where we predict individuals who maximize the considered k-dimensional vectors of evolution indices as rising stars. Let $f_1, f_2, \ldots, f_k$ be the objective functions to be maximized. $\vec{X_1}$ dominates $\vec{X_2}$ if $f_i(\vec{X_1}) > f_i(\vec{X_2})$ for some f_i, and $f_j(\vec{X_1}) \geq f_j(\vec{X_2})$ for all f_j. Assume a set of solutions $X = \{\vec{X_1}, \vec{X_2}, \ldots, \vec{X_m}\}$, $\vec{X_i}$ is *non-dominated* if no $X_j \in X$ dominates $\vec{X_i}$.

In our problem, $\vec{X_i}$ is the vector of $k-$dimensional evolution indices of key player scores over k data sources. Hence the non-dominated set consists of a set of users who possess an evolution score better than everyone else in at least one of the data sources.

Often, another question of interest is: *Rank the first N rising stars*. MOO approach only returns a set of rising stars, without assigning rising star scores. To answer this question, we have used the rank aggregation method.

2.3 Methodology for Academic Domain

We begin with the academic domain to illustrate the general methodology. Later, we show how to generalize the algorithm to other domains using Information Security StackExchange Q&A forum data.

We have considered co-authorship data, citation data and publication venue data for the analysis. Brief descriptions of each of these networks are presented in Sections 2.3.1, 2.3.2 and 2.3.3. Any centrality measure (such as Eigenvector centrality, Closeness centrality, etc.) can be used to assign key player scores. The only requirement is that the chosen centrality measure represents the importance of a node in the network. Liu et al. [64] and Ding et al. [28] suggest using the PageRank algorithm to find key players in coauthor networks and citation networks, respectively. We have also applied the PageRank algorithm to determine key players of citation and co-authorship networks. For publication venue networks, we have used the publication venue score proposed by Li et al. [61] to find key players.

For each author, the least squares regression method is applied on key player scores, and the associated gradient is used as the evolution metric. The MOO approach uses three criteria to find rising stars:

1. Co-authorship network PageRank gradient
2. Citation network PageRank gradient, and
3. Publication venue score gradient.

2.3.1 Co-authorship Network

$coauthor_i =< N_i, E_i >$ for $i = 1, \ldots, T$ is a weighted directed graph where N_i represents the set of nodes (authors) and E_i represents the set of edges in the network. An edge from node u to v represents that authors u and v have co-authored a publication. The weight $w(u, v)$ associated with edge (u, v) is calculated as

$$w(u, v) = \frac{|pub(u) \cap pub(v)|}{|pub(u)|} \tag{2.1}$$

where $pub(u) = \{p \mid p$ is a publication of $u\}$ and $pub(u) \cap pub(v)$ represents common publications of u and v. Usually, co-authorship networks are modeled as undirected graphs. In the rising star problem, Li et al. [61] proposed using a directed graph to represent influence between two researchers. Following this argument, we have modeled the co-author network as a directed graph, where edge weights represent proportion of publications co-authored together.

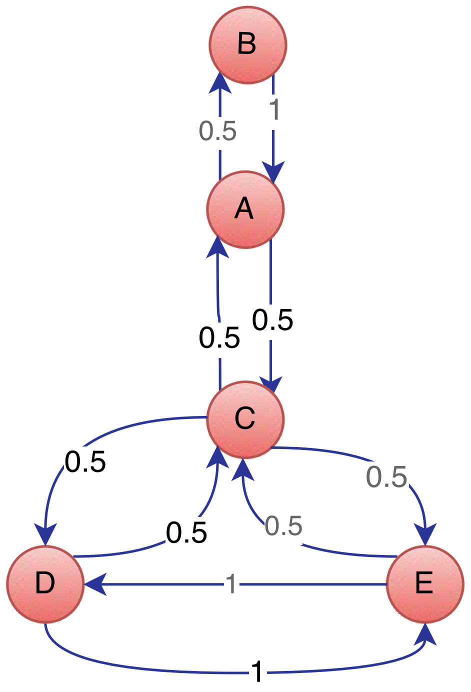
Consider the example given in Table 2.1. Considering coauthor relationships, the weights assigned to links are: $w(A, B) = 1/2$, $w(A, C) = 1/2$, $w(B, A) = 1$. Figure 2.1a shows all weight assignments.

Paper	**Authors of the paper**	**Cited papers**	**Cited papers without self-citation**	**Authors of cited papers (without self-citation)**
1	A,B	-	-	-
2	A,C	-	-	-
3	C,D,E	1,2	1	A,B
4	D,E	1,2	1,2	A,B,C

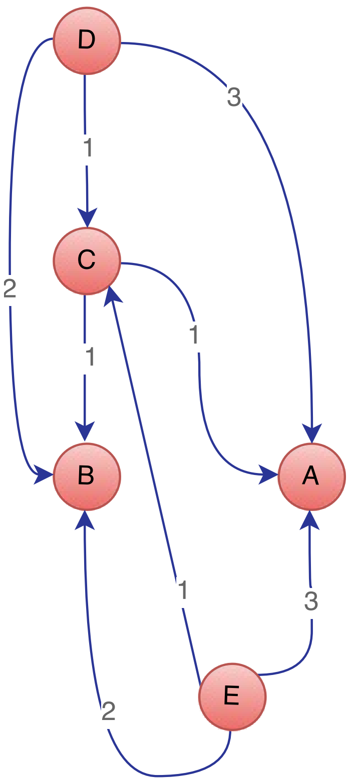
Table 2.1: Co-author and paper citation network data

2.3.2 Author Citation Network

The author citation network, $citation_i =< N_i, E_i >$ for $i = 1, \ldots T$, is a weighted directed graph where N_i represents the set of nodes (authors) and E_i represents the set of edges in the network. An edge $(u, v) \in E_i$ represents a citation of author v by u, and the weight of edge (u, v) refers to the number of citations of v by u. If two papers have common authors and if there is a citation of one paper by the other, we have considered them as self-citations and ignored them when building the network. Figure 2.1b shows the corresponding citation network for paper citation data in Table 2.1.



(a) Directed co-author network weight assignment



(b) Directed author citation network structure

Figure 2.1: Co-authorship and author citation network examples

2.3.3 Author Publication Venue Network

The publication venue network is a weighted undirected bipartite network, $venue_i =< N_i, E_i >$ for $i = 1, \dots T$, where nodes include authors and publication venues. An edge $(u, v) \in E_i$ between u and v indicates that author u has published in venue v at time t_i. The weight of the edge $w(u, v)$ represents the number of publications of author u in venue v. All publication venues were categorized into three ranks based on Arnetminer venue impact scores. The top ranked 100 venues, according to impact score, are assigned rank = 1, the next 200 are assigned rank =

2 , and all other venues are assigned rank = 3. Since we are interested in publications in top ranked conferences/journals, we only considered publication venues ranked 1 and 2 in the network formulation. Hence nodes in the venue network contain all authors and all publication venues ranked 1 and 2.

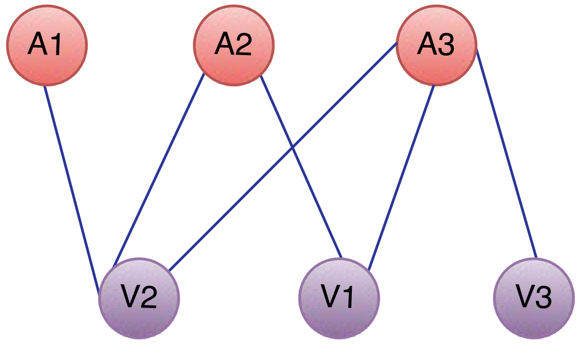


Figure 2.2: Publication venue network

The importance of an author in the venue network is quantified using publication venue scores proposed by Li et al. [61] using equation 2.2. Venue score of an author u, $VS(u)$ at a given time step is calculated using publication venue network at time t_i.

$$VS(u) = \sum_{v \in Neighbors(u)} w(u,v)[\alpha^{(rank(v)-1)}]^{-1} \tag{2.2}$$

In Equation (2.2), α is the damping factor, with a typical value of $\alpha = 2$. For illustration, consider the venue network shown in Figure 2.2, where A1, A2, and A3 represent three authors and V1, V2, and V3 represent three publication venues and their ranks are 1, 2, and 1 respectively. The venue scores of the authors are as follows.

$$VS(A1) = 2.5, VS(A2) = 1.5, VS(A3) = 2.5.$$

2.3.4 PageRank Algorithm

To calculate influence in a network, PageRank algorithm has been widely used. Even though the original purpose of PageRank algorithm was to rank web pages based on link relationships, it can be effectively used to rank nodes in a graph, given that important nodes often tend to interact with other important nodes [28, 64].

PageRank of a node u is obtained using equation 2.3.

$$PR(u) = \frac{(1-d)}{N} + d\sum\nolimits_{v \in In(u)} \frac{PR(v).w(v,u)}{\sum\nolimits_{k \in Out(v)} w(v,k)} \tag{2.3}$$

where $In(u)$ represents set of incoming edges to node u and $Out(u)$ is the set of outgoing edges from u.

The number of authors in coauthor and citation networks typically increase at each time step. On the other hand, scores assigned by the PageRank algorithm depend on the number of nodes in the network. Thus, PageRank scores of authors tend to decrease over time. In order to account for this effect, when calculating evolution index of each node, we have considered the *rank of PageRank score* of each author instead of the actual PageRank value.

2.4 Proposed Algorithm

The proposed algorithm models academic data in three different networks, considering three aspects: co-authors, citations and publication venues. All of these aspects are important for an author to be a rising star. A rising star in academic domain would collaborate with other important researchers and will also be cited

Algorithm 1 Find Rising Stars

```
Input: Bibliography data set including publications, citations and publication venues,
    Time period $t = 1, 2, \ldots, T$.
Output: $RS$: Set of rising stars
1: for $u \in Authors$ do
    (a) Rank of PageRank score of author $u$ is obtained for each co-authorship network.
        Let the PageRank score ranks be $P_1^{(1)}(u), P_2^{(1)}(u), \ldots, P_T^{(1)}(u)$
    (b) Rank of PageRank score of author $u$ is obtained for each citation network. Let
        the PageRank score ranks be $P_1^{(2)}(u), P_2^{(2)}(u), \ldots, P_T^{(2)}(u)$
    (c) Venue score of author $u$ is obtained for each publication venue network. Let
        venue scores be $P_1^{(3)}(u), P_2^{(3)}(u), \ldots, P_T^{(3)}(u)$
    (d) Perform Linear Regression of the PageRank score ranks and venue scores over
        time to determine the slopes. Let the associated slopes be $S^{(1)}(u)$, $S^{(2)}(u)$ and
        $S^{(3)}(u)$ respectively.
2: $RS$ = Following MOO approach or rank aggregation method find set of rising stars
    considering $\left(S^{(1)}(u), S^{(2)}(u), S^{(3)}(u)\right)$ for all authors
```

by other important researchers. Using PageRank to rank authors (in co-author and citation networks) captures this idea. Since the work of a rising star should be promising, they should publish in good conferences/journals. Considering these three networks separately allows us to observe an author's profile in all three aspects. Further for someone to become a rising star they should show an improving profile over time. Taking regression slope accounts for the evolution factor.

2.4.1 Multi-Objective Optimization Approach to Find Rising Stars

Combining Rising Star scores from each data type can be modeled using dominance relation in multi-objective optimization. In our academic network scenario we have three objectives; co-author network slope, citation network slope and venue score slope. We want to find a set of users, who outperformed other users in at least one of the given objectives. In other words, a rising star discovered by the MOO approach for academic data would have outperformed all others in at least one of the objectives.

2.4.2 Rank Aggregation to Find Rising Stars

In the MOO approach all the predicted rising stars will have an equal importance. However, often we desire to have rankings of the rising stars. This can be accomplished by rank-aggregation approach. Algorithm 2 explains how we calculate a rising star score for each author in bibliography data where Borda's method was used for rank aggregation. Rank assignment in Borda's method is as follows. If there are N individuals to rank, we assign N points to the one with highest value, $N-1$ to next highest ranked and so on.

In a domain where we can define relative importance of each objective for rising star calculation, we can effectively use rank aggregation to assign weights to each objective. Now we can take a weighted sum of the objective scores as the rising star score of each individual. Given k heterogeneous networks, key player evolution index $S^1, S^2, \ldots, S^k$, weight assignment to objectives as $w_1, w_2, \ldots, w_k$, the Rising star score of an author u can be calculated using equation 2.4. The $score(S^i(u))$ is the rank score assigned to $S^i(u)$ following Borda's method.

$$RisingStarScore(u) = \frac{\sum_{i=1}^{k}[w_i \times score(S^i(u))]}{\sum_{i=1}^{k} w_i} \tag{2.4}$$

Assigning weights to each objective depends on the data set and the key player criteria we use. Liu et al. [65] propose a supervised learning algorithm to assign weights to each objective. Their method minimizes the difference between ground truth ranking and ranking according to each objective.

Algorithm 2 Rank Aggregation Algorithm

```
Input: Co-author network PageRank slopes ($S^{(1)}$), Citation network PageRank slopes($S^{(2)}$) , Venue Score slope ($S^{(3)}$), Weights for each objective $w_1, w_2, w_3 > 0$.
Output: Rising star scores for each author
1: for $u \in Authors$ do
2:     Find Rank Score for co-author PageRank slope $S^{(1)}(u)$. Let $S^{(1)}(u)$ rank score be $Score(S^{(1)}(u))$.
3:     Find Rank Score for citation PageRank slope $S^{(2)}(u)$.Let $S^{(2)}(u)$ rank score be $Score(S^{(2)}(u))$.
4:     Find Rank Score for venue score slope $S^{(3)(u)}$. Let $S^{(3)}(u)$ rank score be $Score(S^{(3)}(u))$.
5:     Rising_Star_Score($u$)=$\frac{\sum_{i=1}^{3} w_i * S^{(i)}(u)}{\sum_{i=1}^{3} w_i}$.
```

2.5 Experimental Evaluation

Co-authorship networks, author citation networks and publication venue networks were generated using ArnetMiner bibliography dataset [99] for each year from 1990 to 1995. There were 30,586 unique authors in all 6 networks. When formulating the citation network, self citations were eliminated by removing any paper citation containing common authors. Top 1000 ($\approx$ 3%) ranked authors in citation and coauthor networks in 1990 were removed considering they are already stars in the network. We focus only on authors who have the potential to be considered as “stars" in near future.

To evaluate the outcome of our algorithm, each predicted rising star is evaluated in 2006 and also in 2014 using authors’ H-index, total number of papers, and total number of citations. We have extracted their performance in each metric in 2006, using Arnetminer bibliography data. For 2014 we have considered their profiles in Google Scholar and Arnetminer.

	2006			2014		
Author ID	**H-index**	**Papers**	**Citations**	**H-index**	**Papers**	**Citations**
1	19	141	1267	94	646	60904
2	11	33	550	41	104	14090
3	24	86	2438	109	247	73772
4	23	100	1933	92	249	43833
5	10	93	347	59	269	13261
6	12	47	397	46	204	9616
7	16	76	762	69	208	21685
8	13	166	640	65	475	17231
9	15	42	554	42	83	8558
10	9	44	290	37	136	4285
11	6	65	140	24	152	1923
12	20	53	1288	64	165	55717
13	5	64	79	42	232	10740
14	15	88	1043	60	192	17615
15	13	99	601	59	294	11638
Average	**14.06**	**79.8**	**821.93**	**60.2**	**231.53**	**24324.53**

Table 2.2: Rising stars discovered using Multi-Objective Optimization approach

2.5.1 Multi-Objective Optimization Approach Results

Based on 1990-1995 data, MOO approach predicted fifteen non-dominated authors as rising stars. Almost all (except one) individuals identified as rising stars by our MOO algorithm did indeed become successful, as shown by Table 2.2.

In 2006, the average values of H-index, number of papers and citations by our method were 14.06, 79.8 and 821.93. In 2014 all three metrics increased significantly to 60.2, 231.53 and 24324.63. In particular the significant increase in the H-index (in 2014) indicates that the authors continued to rise in academic reputation.

Figure 2.3 shows how average citations change for all authors from 1996 to 2012 using their Google Scholar profiles. According to Figure 2.3, the rising stars have continued to improve their profiles and have stabilized their profiles as stars in 2007.

We have compared the performance of our algorithm with the algorithm proposed by Li et al. [61] since other solutions are minor variations of Li et al. [61] pro-

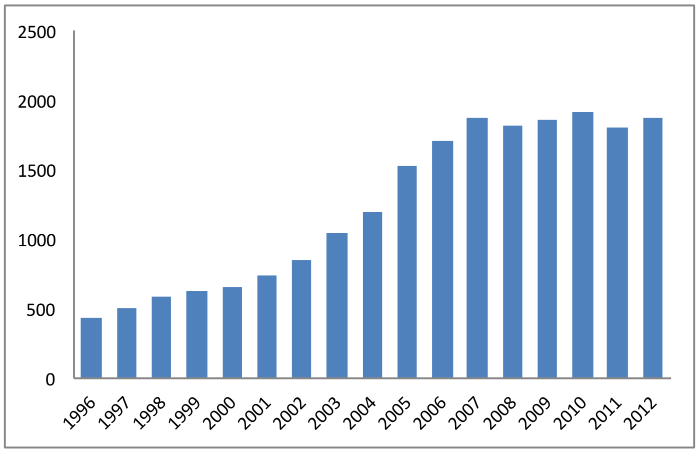


Figure 2.3: Average citations in each year for all MOO approach rising stars with Google Scholar profiles

posed methodology. Recall that these authors use weighted PageRank algorithm on a weighted co-authorship networks together with a node weight assigned using publication venues to predict rising stars. Li et al. [61] predicts any author with PageRank gradient in top 90th percentile as a rising star. Since our MOO approach returns 15 unique individuals as rising stars we have compared performance of MOO approach with top 15 authors with highest PageRank gradients from the method proposed by Li et al. [61].

Figure 2.4 shows the comparisons of average H-index, average number of papers and average citations. Average H-index, average total number of papers and average total citations of the rising stars obtained using their algorithm are much smaller than those obtained by our approach, clearly implying superiority of our method. Differences in averages were evaluated using the t-test, and found to be statistically significant.

Figure 2.5 shows the performance of the MOO approach compared to co-author PageRank slope, citation PageRank slope and venue score slope separately. Rising stars identified using MOO approach performs well in all three measures compared to rising stars from each network separately.

Meho et al. [72] states that an “outstanding scientist" will have a H-index of

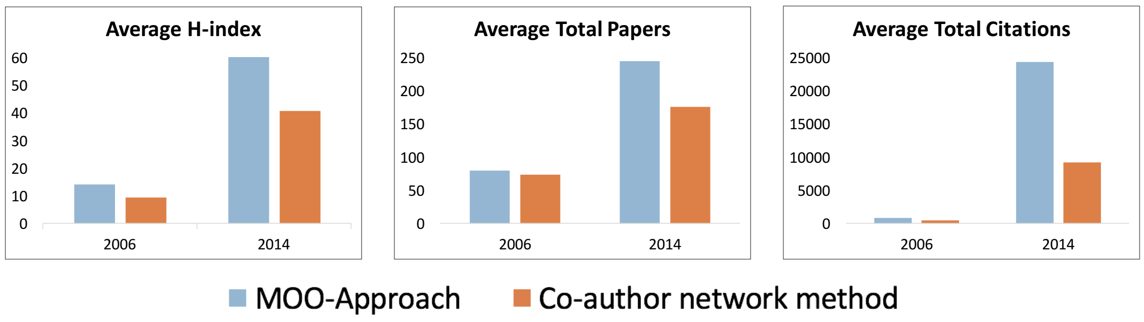


Figure 2.4: Comparison of MOO approach and co-author network method proposed by Li et al. [61]; Average H-index of each rising star, Average number of Papers of each rising star, and Average number of citations for each rising star respectively

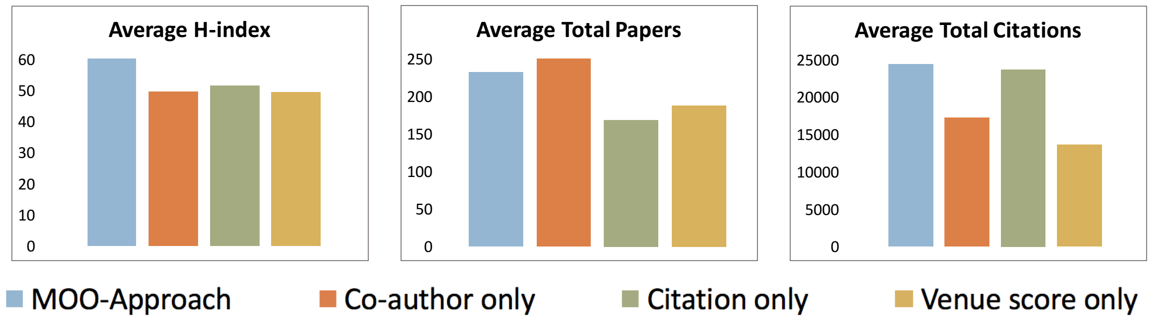


Figure 2.5: Performance of MOO approach compared to using co-author PageRank slope, citation PageRank slope, venue score slope separately

40 and a “truly unique" individual will have H-index of 60. As seen in Table 2.2, all except one predicted rising stars have become outstanding researchers by 2014. According to the study conducted in [32] on average a well established author will have 56.79 papers, 347.86 citations and an average H-index of 14.31. The rising stars predicted by our approach are truly outstanding with an average of 231 papers and 24324 citations. Performance of the predicted rising stars continued to grow in all three metrics, as seen by comparing 2006 and 2014 data, supporting our algorithm further.

2.5.2 Thin Layer of Non-dominated Individuals

Results in section 2.5.1 only consider the non-dominated individuals with respect to three objectives: co-author PageRank slope, citation PageRank slope and venue

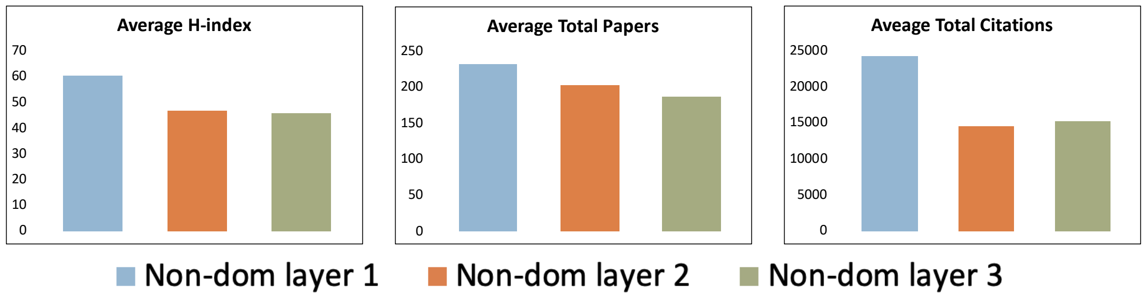


Figure 2.6: Performance of first three layers of Non-dominated rising stars

score slope. But there can be individuals who are dominated by a small margin and hence not appearing among rising stars. In order to account for these individuals, we removed the first set of rising stars and again calculated the non-dominated set. We continued this for three iterations. Figure 2.6 shows performance of predicted rising stars. The first layer of non-dominated authors performs significantly better than second and third layer. The difference between the second and third layers is minimal.

2.5.3 Rank Aggregation Approach Results

Table 2.3 shows the profiles of top 15 predicted rising stars based on 1990-1995 Arnetminer bibliography data using rank aggregation method. Here we have considered that all the objectives are equally weighted. Results show that predicted rising stars have indeed become “Outstanding researchers" by 2014, as measured in terms of H-index.

We present rank aggregation method as an alternative method to find rising stars, when we have to assign rank to each rising star. Results in Table 2.3 show that performance of this method is comparable to the MOO approach, slightly worse but requiring less computation.

	2006			**2014**		
Author ID	**H-index**	**Papers**	**Citations**	**H-index**	**Papers**	**Citations**
8	13	166	640	65	475	17231
9	15	42	554	42	83	8558
12	20	53	1288	64	165	55717
7	16	76	762	69	208	21685
14	15	88	1043	60	192	17615
16	13	71	521	57	207	12995
17	10	219	360	45	425	9062
18	12	152	689	69	292	19041
19	15	35	825	42	96	13108
20	10	135	456	58	270	12784
21	13	106	801	79	251	23207
5	10	93	347	59	269	13261
22	10	50	511	41	116	12208
23	18	83	839	92	587	44735
24	13	65	751	41	202	7494
Average	**13.53**	**95.6**	**692.46**	**58.86**	**256.87**	**19246.73**

Table 2.3: Profiles of top ten rising stars obtained by using the rank aggregation approach

2.6 Finding Rising Stars in Q&A Forums

To generalize the rising star finding algorithm to another domain, we used Information Security Stack Exchange[50] question answer forum data to find rising stars in the forum. In this Q&A forum there are three possible types of interactions that can exist among users, shown in Figure 2.7. A user can answer a question posted by another user (QA), a user can comment on a question posted by another user (QC) and a user can comment on an answer provided by another user (AC). We consider the network consisting of these three types of interactions as a heterogeneous network.

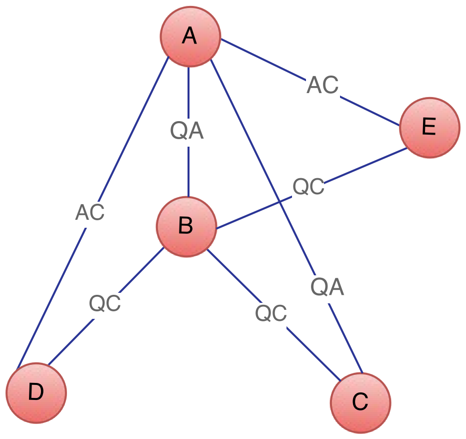


Figure 2.7: Stack Exchange Q&A forum heterogeneous graph structure

2.6.1 Network Formulation

To represent each type of interaction, we have built three different networks for Q&A forum. In each network we used PageRank algorithm to assign an importance score to each node.

Question-Answer Network

Question-Answer network, $G(V, E)$ is weighted, directed graph, where $V =$ users of the forum and an edge $(u, v) \in E$ if v has answered $u's$ question. We have considered the reversed direction of the Question-Answer relationship as the direction of the edge. The intuition behind this comes from PageRank algorithm perspective. An important individual should have more incoming edges than outgoing edges. If v has answered $u's$ question, this implies that v is more important in the network than u, hence we create an edge from u to v. The weight of edge (u, v) represents how many of $u's$ questions were answered by v.

Question-Comment Network and Answer-Comment Networks

Question-Comment and Answer-Comment networks, $G(V, E)$ are weighted, undirected graphs, where $V =$ users of the forum and an edge $(u, v) \in E$ if v has commented on $u's$ question or answer. Here also the weight represents the number of questions/answers the two users have commented on each other. We have formulated these two networks as undirected since a comment does not necessarily say who influences whom. Further a comment can be positive or negative.

2.6.2 Results and Evaluation

We have collected data for Information Security Stack Exchange forum from 2012-2014. For each year, we built a network for each interaction type. After calculating PageRank scores for all nodes in each network for each year, we used the ranks of the PageRank values to find the evolution index for a node in a given network. The slope of the regression line fitted to PageRank value ranks is considered as the evolution index. We removed the top 1000($\approx 10\%$) users with highest PageRank scores in 2012 Question-Answer network considering they are already experts in the network.

The found rising stars are evaluated using the number of answers a rising stars has given in 2015, the number of answers accepted as the correct answer and the reputation score of the rising star at the end of 2015. The reputation score is a community given vote in the Stack Exchange data, which represents the acceptance of a user in the Q&A forum.

Table 2.4 shows the average number of questions answered by found rising stars, the average number of answers accepted for a rising stars in 2015 and the average reputation. Results show that rising stars found using our algorithm 1 are significantly different other users. Note that these users were not among top

	Average number of answers given in 2015	Average number of answers accepted in 2015	Average reputation
MOO approach (3 rising stars)	56.3	20	17456
Rank Aggregation (Top 10)	56.6	21.5	11187
Average for a normal user	4.02	1.06	241

Table 2.4: Q&A forum rising stars results compared with a normal user

User name	No. Answers given in 2015	No. Accept Answers 2015	Reputation	StackExchange Given Rank (2016)
Philipp	103	28	22467	Top 0.27%
Mark	88	29	21923	Top 0.79%
RoraZ	63	29	7978	Top 2%

Table 2.5: Rising stars found using MOO approach for Q&A forum data

10% of the users with highest PageRank values in 2012, but still they manage to perform well above an average user.

Even though the MOO approach has only identified three users as the rising stars in Q&A forum, results in table 2.5 shows that all the identified individuals are truly outstanding, by being ranked among the top 2% of all the information security Stack Exchnage users in 2016.

Table 2.4 shows that the top 10 ranked rising stars found using unweighted rank aggregation algorithm gives similar results to the MOO approach in Stack Exchange data. In addition, the rank aggregation algorithm also finds a set of users significantly different from most other users.

2.7 Conclusion

In this chapter, we present an algorithm to find rising stars in heterogeneous dynamic social networks. Our algorithm first identifies possible rising stars from each data source separately and uses a MOO approach and rank aggregation approach to find rising stars by considering all data sources.

The proposed algorithm was able to identify a set of rising stars in bibliography data, who in turn became outstanding researchers in a few years. On bibliography networks our technique clearly shows superior performance over existing methods. Using Information security StackExchange data, we have shown how the algorithm can be applied to other scenarios. In Q&A answer forum also, our method performs well to identify a set of users, who outperform most other users.

Even though the proposed rising star prediction algorithms can guide identifying most influential people, star researchers or experts in the future, crucial decisions such as hiring an individual should not be solely based on such measures. For example, when identifying rising stars in academia, we consider the importance of an individual's collaborators. If an academic attends a prestigious university, she may get a better opportunity to work with expert researchers compared to others. The proposed algorithms cannot identify such biases. Therefore, these measures should be used with human supervision while considering other factors that would make someone an expert in a given field.

CHAPTER 3 SAMPLING SOCIAL NETWORKS TO LOCATE PEOPLE OF INTEREST

In Chapter 2, we proposed the MOO algorithm to find rising stars in social networks considering multiple types of networks and dynamic nature of the networks over time. The purpose of the work in this chapter is to extend this problem to locate POIs in networks when POIs try to disrupt the identification by providing misleading information and altering network structure to make POI properties not correlated with the network structure. One such example is analysis of so-called “dark networks" representing illegal, covert, or undisclosed activities [84], where individuals within the network may purposefully conceal their network data with the goal of misleading the data collector or analyst.

In this Chapter, we develop algorithms for sampling dark networks with the intention of locating as many POIs as possible in the network given a limited query budget. Here, a POI is a node possessing a certain attribute (e.g., individuals involved in a particular criminal action). As an example, one of the networks we consider contains data depicting relationships among terrorists in the “Noordin Top" Network in Indonesia. In this network, edges represent that two nodes belong to the same organization, attended the same school or training, have a kinship relationship, and so on.[1] In such a network, the POIs may be those individuals who have an attribute of interest to the analyst, such as involvement in a terrorist attack. We assume that we begin with knowledge of one POI in the network, with the rest of the network unobserved (both in terms of topology as well as node attributes).

A major complicating factor in this problem is that due to the covert nature of the networks being studied, one cannot expect the observed information to be reliable. When exploring a dark network, nodes may deliberately provide misinformation about themselves and the network structure (e.g., lie to an investigator), or the analyst herself may need to draw conclusions from multiple sources of information, with possible errors. This is in contrast to traditional sampling algorithms, which generally assume that the information observed is correct at the time of the query. There are thus major challenges in designing algorithms to sample dark networks: first and foremost, in order to select the next query, one must draw inferences about the network structure from inaccurate data. Even for already-observed nodes, when predicting whether a node is or is not a POI, one can only use information observed so far; but because this information may be incorrect, one may come to a false conclusion, which in turn affects predictions for other nodes, potentially leading to a cascade of errors.

In this chapter, querying a node refers to collecting data about the node, node's neighbors, whether these neighbors are POIs and determining whether the queried node is a POI itself. For example, querying a node can refer to a data analyst obtaining information about the node and determining whether it is a POI. The role of the sampling algorithm is to suggest a query on the node that is most likely to be a POI.

We consider two sampling settings, corresponding to different types of data

[1]The data was collected by [86] and compiled into a network by [38].

collection processes. In the first setting, when a node is selected for query, then (1) the data collector accurately discovers whether or not that node is a POI, and (2) the data collector, possibly inaccurately, determines the identities of node's neighbors and whether those neighbors are POIs. In the second setting, we remove the assumption that the data collector can accurately discover a node's status as a POI: instead, the query may misreport whether or not the node itself is a POI. In both settings, there may be errors in determining the queried node's neighbors and whether those neighbors are POIs: for example, a POI node may encode communications with other POI nodes, or try to make these communications look innocent. The first setting corresponds to cases in which it is easy to determine whether a queried node is of interest (e.g., comparing fingerprints to those found at a crime scene). The second setting corresponds to cases in which the determination of whether a node is a POI is difficult and may be done incorrectly (for example, by an analyst aggregating several sources of conflicting information).

We present two sampling algorithms, `RedLearn` and `RedLearnRS`. `RedLearn` operates under the assumption that the analyst can accurately determine the queried node's status. For the second sampling setting, we introduce `RedLearnRS`, a *re-sampling* algorithm that may repeatedly query the same node in order to update the accuracy of the information. We show that in cases where the POIs exhibit homophily (i.e., are likely to be connected to other POIs), a simple algorithm of choosing the node with the most POI neighbors works well. However, in the more realistic scenario where POIs hide their connections with other POIs, REDLEARN and `RedLearnRS` show outstanding performance, improving over the best baseline algorithm by up to 340%.

3.1 Problem Definition

We assume that there is an unobserved, undirected, unweighted graph $G = (V, E)$, in which each node $v \in V$ has a color attribute, $c_v \in \{red, blue\}$. Red nodes represent the persons of interest (POIs) and blue nodes represent all others.

We begin with the realistic assumption that a red node was identified in G (e.g., arrested while committing a crime). We are given a budget b of queries, where each query is conducted by analyzing data corresponding to a node. We call this process placing a *monitor* on a node.[2] In each step, we query an observed node v, regardless of its color, and the monitor reports (1) the suspected color of the node (s_v), (2) the neighbors of the node ($N(v)$), and (3) the color e_u of each of node v's reported neighbors u. As monitors represent humans performing data analysis, some or all of this information may be reported incorrectly by the monitor, due to analyst error or judgment.

There are four possible types of errors when retrieving node color and network structure information using monitors: (1) A *node color error*, where the suspected color of the node is not the true color ($s_v \neq c_v$) (2) an *edge existence error*, where the monitor fails to report some neighbors of the node, (3) an *edge non existence error*, where the monitor reports false edges between nodes and (4) a *neighbor relationship color error*, where the reported colors of the queried node's neighbors are different from their true colors ($e_u \neq c_u$).

For example, placing a monitor on a suspected criminal node could represent an analyst conducting an investigation to determine whether this person is a criminal (node color). The investigation process includes looking through his email and phone contacts to observe his connections (neighbor identities), analyzing his emails and other messages with these contacts to determine whether these neigh-

[2]This terminology was motivated by an application in which hardware devices known as monitors are placed on computers to observe incoming and outgoing traffic.

bors are themselves criminals (neighbor relationship colors). Naturally, there may be errors in this process due to nodes trying to hide/fabricate their connections, criminals using code words in their communications, general noisy communications among all nodes and analysis errors that can occur. [3]

Suppose that after exhausting the monitor budget b, we have a set of monitored nodes $V_m = V_r \cup V_b$. Here, $V_r = \{v \in V_m : s_v = red\}$, the set of monitored nodes that are *suspected* to be red and $V_b = \{v \in V_m : s_v = blue\}$, the set of monitored nodes that are *suspected* to be blue. The goal of the analysis is to maximize the true set of red nodes detected, namely $|\{v \in V_r : c_v = red\}|$.

We consider two problem settings depending on whether the analyst can conclusively determine someone is a criminal or not.

3.1.1 Problem Setting 1: Node color reliable

In Problem Setting 1, we assume that a monitor accurately determines the color of the queried node. Therefore, $V_r = \{v \in V_m : s_v = red\} = \{v \in V_m : c_v = red\}$; all nodes that are suspected to be red are indeed red colored nodes.

For example, suppose that there is an investigation to find a set of computers that are affected by some virus. The analyst has a hardware device which can detect whether a computer is affected or not. Whenever the analyst finds a potential affected computer, she can place this device on the machine and determine the true status of the machine. In this example, placing the hardware equipment is equivalent to placing a monitor. Because the analyst has limited hardware equipment, she needs to carefully decide which computer to monitor next so that V_r is maximized.

[3]We consider realistic types of errors to represent analysis errors; these are described in Section 3.5.5.

3.1.2 Problem Setting 2: Node color misreported

In Problem Setting 2, we consider the setting where a monitor may misreport the color of the monitored node. For example, suppose there is an investigation to find terrorists who were involved in a specific attack. The analyst has to determine whether someone is a terrorist solely based on his contacts, messages, and whereabouts. In the absence of conclusive evidence to prove that a person is a terrorist, the analyst has to make a judgment based on available information. This can lead to identifying a terrorist as not guilty if there is not enough evidence; and conversely, may lead to identifying an innocent individual as guilty.

In this problem setting, placing a single monitor on a node may not reveal the true color of a node. In this case, we allow for repeated monitor placement to re-query previously-monitored nodes. In the example above, repeated monitor placement would allow the analyst to gather more information about the criminal by considering other types of communication channels, spending more time and resources on the analysis which can lead to better judgment.

3.2 Background

3.2.1 Criminal Network Analysis

A dark network is a social network representing illegal and covert activities, whose members are actively trying to conceal network information even at the expense of efficiency [6, 84]. Because dark networks are deceptive by nature, there are many errors involved in collecting and analyzing data from these networks.

Criminal networks are a common example of dark network [6]. Some researchers have used social network analysis techniques such as identifying key players, community detection and network visualization techniques to identify leaders of criminal networks, identify subgroups in criminal networks, understand common properties of criminal networks and visualize criminal networks respectively [20, 67, 56, 112]. Some other researchers have considered using social network analysis to disrupt criminal networks [92, 89]. These analysis techniques generally assumes that the complete network structure information is available.

In this work, we introduce a methodology to identify POIs in dark networks, while examining various error scenarios based on the terrorist actively trying conceal their true status and connections. Uncovering these networks amid concealed and misinformation require advanced network sampling techniques.

3.2.2 Network Sampling Algorithms

In general, network sampling algorithms can be divided into two categories: downsampling algorithms and crawling based sampling algorithms. Downsampling algorithms begin with knowledge of the entire graph, but due to computational issues (e.g., time or space), must find an appropriately-sized subgraph that is representative of the larger graph [59]. Crawling based algorithms on the other hand, start with knowledge of few nodes and explore the network through querying observed nodes. In our problem, we use crawling to explore the network structure by placing monitors.

There are a multitude of sampling techniques for crawling based network exploration, including random walks [4, 49, 77], biased random walks [35], and walks combined with reversible Markov Chains [2], Bayesian methods [34], or standard exhaustive search algorithms like depth-first or breadth-first searches, such as [1, 9, 10, 24, 59]. However, these methods do not use all discovered information, such as node attributes of the discovered part of network, and are not designed for sampling networks in which queries return inaccurate information.

Various researchers have considered the problem of sampling for specific goals.

For example, [5] present an algorithm to sample the node with the highest estimated unobserved degree. [43], and [69] examine online sampling for centrality measures. [68] develop a guilt-by-association method to identify suspicious individuals in a partially-known network.

Some researchers have considered the problem of sampling to locate targeted nodes. [11] present a volatile multi-arm bandit based algorithm to crawl for targeted nodes. Their approach tries to find a balance between exploring the network and exploiting targeted nodes at the time of crawling. However, this work does not address potential inaccuracies in the sampling process (and additionally, using an explore-exploit method for sampling dark networks would present significant ethical problems, as it would require investigating non-suspects for the sake of exploration). [93] present the TONIC problem for finding targeted nodes, and propose a method for finding nodes that can lead to the targeted nodes. Unlike our problem setting, these works do not consider the case where queried nodes can provide information about their neighbors, and also do not account for possibly inaccurate information and errors. [38] consider identifying a target community from partial information, by taking a different approach one what part of the network has been discovered. They assumed that the information of one layer (one type of relationship) of network was known, and tried to identify a group in the whole network.

3.2.3 Node Classification Algorithms

A natural approach for the problem posed in this chapter is to use a node classification algorithm to predict whether observed nodes are red or blue. Traditional data classification algorithms assume that data instances are independent from each other. In contrast, node classification algorithms study the problem, how to conduct a classification in network or relational data. These algorithms work with partially labeled network data to predict labels for unlabelled data instances. Node classification algorithms generally achieve better performance by exploiting relational information between classification data instances.

There are many applications of node classification algorithms. [111] showed applications of collective classification in classifying gender, political views, religious views and relationship status using Facebook data. [115] showed that privacy in social media sites can be exploited with a few public accounts using collective classification. [17] used node classification algorithms to improve performance of sentiment classification in congressional floor debates. [19] showed that collective classification approach can be successfully used to classify whether an email has an action request based on sequence of emails in a thread.

There are two main categories of node classification algorithms [8]. The first category uses a local classifier trained using known labels and network structure information to predict node labels. Iterative Classification Algorithm (ICA) [76] and structured logistic regression model are such examples, and both use local information and links [66]. [37] proposed to use dummy edges to allow information flow in networks and looking at second neighborhood of a node instead of first neighbors to improve ICA performance.

The second type of node classification algorithms are the label propagation based algorithms [117, 63]. These algorithms use random walks to learn a global labeling function. Label propagation based algorithms do not require neighborhood based features to predict labels. They tend to explore the inherent link structure. Both types of algorithms assumes some type of association between nodes with same labels.

In our problem setting, we have incorporated node colors, potential colors of neighbors as well as link structure to identify POIs. Label propagation based algorithms only explore the link structure, and so are not capable of using neighbor color information given by monitored nodes. ICA is the most widely used algorithm among local classifier based node classification algorithms. ICA uses a traditional classification algorithm as the local classifier, which is trained using the labeled nodes in the network. The unlabeled nodes are labeled according to the predictions from this local classifier. ICA then iteratively recalculates features for nodes, using these predicted labels and predicts labels again. This process continues until the predicted labels converge. In Section 3.6.1, we present how ICA performs in finding POIs.

3.3 Proposed Method

We propose algorithms `RedLearn` and `RedLearnRS` for Problem Settings 1 and 2, respectively. In Section 3.3.1, we present `RedLearn`, a novel re-sampling based algorithm to find POIs, where the node colors reported by the queries is accurate. In Section 3.3.2, we present `RedLearnRS`, an extension of `RedLearn` for the more general Problem Setting 2, where the node colors returned by queries can be wrong. These two algorithms use features that include node color information, estimated neighbor color information, as well as network topology. Both of these algorithms operate under possible edge existence errors and neighbor color errors.

3.3.1 `RedLearn`: A Learning Based Monitor Placement Algorithm

We introduce `RedLearn`, a learning-based sampling algorithm for finding nodes of interest under Problem Setting 1 (i.e., the reported node color is the true node color). `RedLearn` uses nodes' structural features and stated attributes to predict labels for unmonitored nodes, as described in Table 3.1.

To understand the use of network structural features in predicting node labels, consider the following: Intuitively, if the original network displays node color homophily, the probability of node v being a red node is higher if it has many red neighbors. However, if the network displays low or even anti-homophily, using this measure will naturally result in poor performance.

For example, some monitor placement algorithms, such as selecting the node that was reported as red by the greatest number of monitors, depend on information retrieved from neighbors. The performance of such monitor placement criteria thus heavily depends on neighbor color errors and edge existence errors.

To overcome these dependencies, the proposed `RedLearn` algorithm models this as a two-class classification problem, but rather than predicting whether a node v is red or blue, we instead predict $P(c_v = R)$.

Features: Table 3.1 describes the set of features used by `RedLearn`. There are two types of features: (a) Network structure-based features $(1, 2, 3, 4, 5)$, and (b) Neighbor-reported color-based features $(6, 7, 8, 9, 10, 11)$.

	Feature	Description
(1)	Number of red neighbors	$\lvert\{u \in N(v) \vert s_u = R\}\rvert$
(2)	Number of blue neighbors	$\lvert\{u \in N(v) \vert s_u = B\}\rvert$
(3)	Number of red triangles if v is red	$\lvert\{u, w \in N(v) \vert$ $(u \in N(w)) \wedge s_u = s_w = R\}\rvert$
(4)	Number of second hop red neighbors	$\lvert\{w \in N(N(u)) \vert s_w = R\}\rvert$
(5)	Number of second hop blue neighbors	$\lvert\{w \in N(N(u)) \vert s_w = B\}\rvert$
(6)	Number of neighbors estimate red	$\lvert\{u \in N(v) \vert (e_{uv} = R)\}\rvert$
(6)	Number of neighbors estimate blue	$\lvert\{u \in N(v) \vert (e_{uv} = B)\}\rvert$
(7)	Number of red neighbors estimate red	$\lvert\{u \in N(v) \vert (e_{uv} = R) \wedge s_u = R\}\rvert$
(8)	Number of red neighbors estimate blue	$\lvert\{u \in N(v) \vert (e_{uv} = B) \wedge s_u = R\}\rvert$
(9)	Number of blue neighbors estimate red	$\lvert\{u \in N(v) \vert (e_{uv} = R) \wedge s_u = B\}\rvert$
(10)	Number of blue neighbors estimate blue	$\lvert\{u \in N(v) \vert (e_{uv} = B) \wedge s_u = B\}\rvert$
(11)	Inferred probability of being red	$P^I(s_v = R)$

Table 3.1: Classification features used by `RedLearn` for a node v with neighbor set $N(v)$. Here, c_v is the true color of v, s_v is the suspected color of v and e_{uv} is u's estimate of v's color.

Network structure-based features are used to learn the patterns of connections between red nodes (e.g., homophily vs. anti-homophily), as previous work shows that algorithms perform differently based on the network studied [107]. Neighbor reported color-based features are intended to learn the relationship between what a monitor reports about its neighbors' colors and the true colors of those neighbors. The second hop features $(4, 5)$ are used to better estimate the red nodes that do not show homophily [37].

Inferred probability of being red:

We formulate four different probabilities to measure how likely it is that an unmonitored node is red, based on the neighbor color estimates given by differently colored monitors. We use the observed neighbor color error information to calculate these probabilities (i.e., once we monitor a node and confirm its color, we can determine whether monitored neighbors of this node have misreported its color). These four probabilities provide us information about whether there is a general pattern of neighbor color misreporting behavior. For example, we may learn whether monitors placed on red colored nodes tend to misreport neighbor colors more frequently. In Equation 3.1, we find the proportion of truly red nodes, when other red nodes have estimated them to be red.

Next we present how we compute the probability of a node v being red/blue given the choices of that neighbor u is either red or blue.

$$P(s_v = R|(s_u = R) \wedge (e_{uv} = R)) = \frac{|\{u, v \in V_m|(s_u = R) \wedge (e_{uv} = R) \wedge (s_v = R)\}|}{|\{u, v \in V_m|(s_u = R) \wedge (e_{uv} = R)\}|},$$
$$P(s_v = R|(s_u = R) \wedge (e_{uv} = B)) = \frac{|\{u, v \in V_m|(s_u = R) \wedge (e_{uv} = B) \wedge (s_v = R)\}|}{|\{u, v \in V_m|(s_u = R) \wedge (e_{uv} = B)\}|},$$
$$P(s_v = R|(s_u = B) \wedge (e_{uv} = R)) = \frac{|\{u, v \in V_m|(s_u = B) \wedge (e_{uv} = R) \wedge (s_v = R)\}|}{|\{u, v \in V_m|(s_u = B) \wedge (e_{uv} = R)\}|},$$
$$P(s_v = R|(s_u = B) \wedge (e_{uv} = B)) = \frac{|\{u, v \in V_m|(s_u = B) \wedge (e_{uv} = B) \wedge (s_v = R)\}|}{|\{u, v \in V_m|(s_u = R) \wedge (e_{uv} = R)\}|}.$$

For each unmonitored node v, we infer the probability that it is red $P^I(s_v = R)$, based on colors of its monitored neighbors and color estimates these monitored neighbors have given about v, using Equation 3.1:

$$P^I(s_v = R) \;=\; \frac{\sum_{u \in N(v)} P(s_v = R | (s_u = R/B) \wedge (e_{uv} = R/B))}{|N(v)|}. \qquad (3.1)$$

Training Data: Suppose that we have placed n monitors so far. The training set then consists of the (correctly) reported colors of the n monitored nodes along with their respective feature values. We use each of the n networks obtained after placing each monitor to train the learning model, to determine where to place the $(n+1)^{st}$ monitor.

Classification Algorithm: RedLearn determines $P(c_v = R)$ for each unmonitored node v. Because it predicts a probability rather than a binary label, RedLearn uses a logistic regression classifier. Furthermore, because the learning model must be updated frequently, this classifier gives an added advantage of faster training.

Placing the Next Monitor (Prediction): Given the placement of n monitors and deciding to place the $(n+1)^{st}$ monitor, REDLEARN calculates a feature vectors for each unmonitored node, and applies the classifier to these feature vectors, giving the probability that each unmonitored node is red. RedLearn selects the node with the highest probability for placing the next monitor. Algorithm 3 summarizes RedLearn.

3.3.2 REDLEARN RE-SAMPLING (RedLearnRS) Algorithm

For the second problem setting, we introduce RedLearnRS by adapting RedLearn. In this problem setting, we assume that a monitor may be incorrect when reporting the color of the monitored node. For example, this setting corresponds to criminal investigations, where an analyst needs to make a judgment as to whether an individual is a criminal based on available information. RedLearnRS assumes that such judgment errors occur primarily on red nodes, who may report to the investigator

Algorithm 3 Learning based monitor placement

```
procedure LEARNING(start,budget)
    S ← Graph
    S.add(start), S.add($N(start)$)                    ▷ Starting node and neighbors
    while budget>0 do
        $V_m$ ← list of monitored nodes in S
        TrainingData ← feature vectors for $u \in V_m$
        Train classifier using TrainingData
        $V_u$ ← list of not yet monitored nodes in S
        for $v \in V_u$ do
            Get feature vector for $v$
            $P(c_v = R)$ ← predict $v$'s probability of red using learning model
        Choose node $v$ with maximum $P(c_v = R)$ from $V_u$
        $budget \leftarrow (budget - 1)$
        Use $v$ as next monitor
```

that they are actually innocent blue nodes, while blue nodes are unlikely to claim that they are actually red.

Due to the potential for node color errors, `RedLearnRS` is a *re-sampling* algorithm, which may place monitors on already-monitored blue nodes in order to better identify their true colors. Re-sampling an already monitored node indicates that the analyst spends more resources looking at other types of communication channels or obtaining additional evidence, which in turn can provide a different judgment of the node's color.

`RedLearnRS` uses a two step cycle. In the first step, `RedLearnRS` identifies monitored nodes with likely node color errors, which are considered for re-query. Additionally, `RedLearnRS` removes these nodes from the training data to help establish better ground truth labels for training data. The re-sampling step is viewed as an outlier detection problem: because only red nodes may report node color errors, all nodes with node color errors are among the set of suspected blue nodes and can be viewed as outliers within this set. Identified outliers are added back to prediction set, which now contains all observed unmonitored nodes and re-sampled monitored nodes.

The second step of RedLearnRS is similar to RedLearn. Once the outlier blue nodes are identified and moved to prediction set, the learning algorithm is applied to predict the probability of each node in the prediction set being red. The node with the highest probability of being red is selected as the next node to be monitored. This node may be either an observed unmonitored node or a previously-monitored outlier node. RedLearnRS's flow is depicted in Figure 3.1.

New Features: Table 3.2 shows the new features introduced to the learning algorithm to identify node color errors. The first feature is the number of used monitors. The more times a node has been monitored, while still being reported as blue, the more likely it is that it is actually blue. The next two features quantify whether the monitor placed on this node tends to report neighbor color errors more often as an indication concealing information.

	Feature	Description
(1)	Number of monitors placed	m_v
(2)	Number of blue nodes lied about	$\lvert\{u \in N(v) \lvert (e_{vu} = R) \wedge s_u = B\}\rvert$
(3)	Number of red nodes lied about	$\lvert\{u \in N(v) \lvert (e_{vu} = B) \wedge s_u = R\}\rvert$

Table 3.2: Additional features used by RedLearnRS for a node v with neighbor set $N(v)$. Here, s_u represents the suspected color of u, and e_{vu} represents the node color that a monitor on node v reports u to be.

Since RedLearnRS selects nodes among the set of reported-blue nodes to re-monitor, there is a potential risk of monitoring innocent blue nodes repeatedly. Re-sampling does not necessarily mean that the monitored node will be arrested and interrogated, but rather indicates that an analyst takes further steps to identify true color of a node. However, we need to take measures to prevent infringing privacy of innocent nodes. In this regard, RedLearnRS limits the number of monitors that can be placed on a single node to prevent excessively monitoring the same node.

Repeated Monitor Threshold: One can view this limit using optimal stopping criteria, which use prior probability information to calculate success or failure of

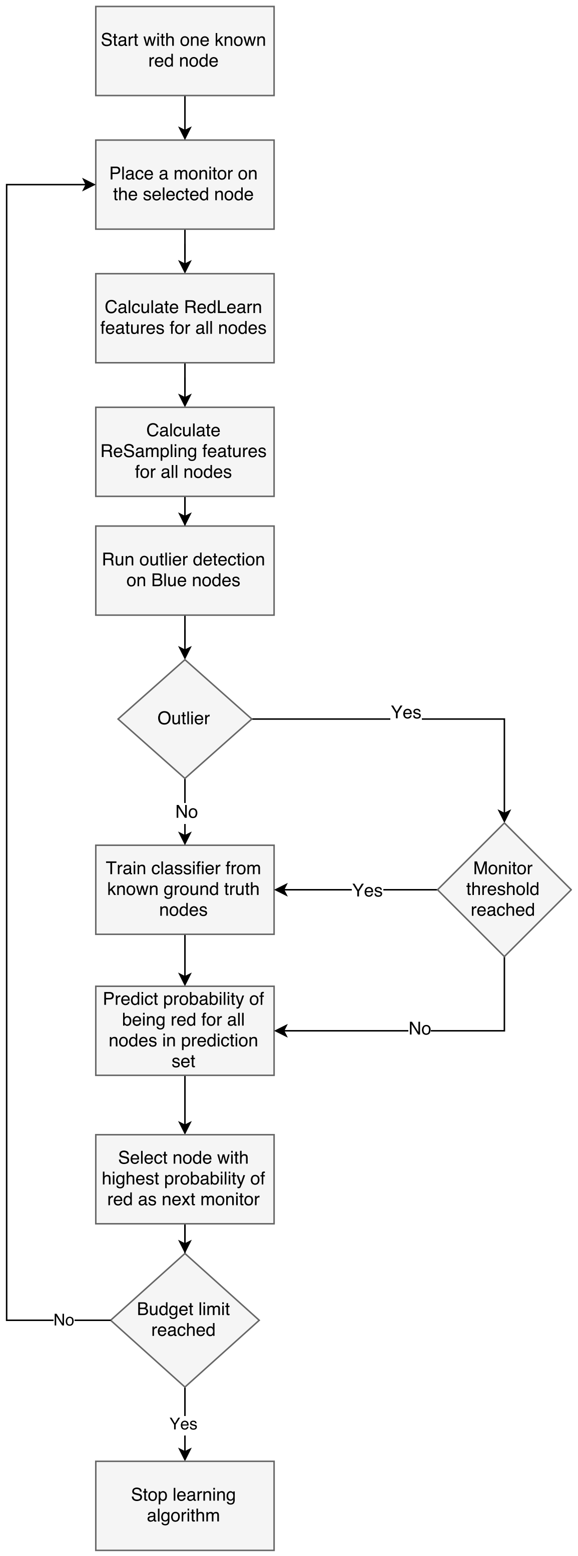

Start with one known red node
Place a monitor on the selected node
Calculate RedLearn features for all nodes
Calculate ReSampling features for all nodes
Run outlier detection on Blue nodes
Outlier
Yes
No
Train classifier from known ground truth nodes
Yes
Monitor threshold reached
Predict probability of being red for all nodes in prediction set
No
Select node with highest probability of red as next monitor
Budget limit reached
No
Yes
Stop learning algorithm

Figure 3.1: `RedLearnRS` algorithm flow. This is a re-sampling based algorithm to identify nodes of interest in a network where node colors reported by monitors may contain errors.

placing another monitor.[4] However, in our case, we would need to repeatedly monitor Blue nodes to estimate the priors required to calculate the optimal stopping point. Instead, `RedLearnRS` uses a simple numerical method to threshold the number of monitors placed on the same node as shown in Equation 3.2.

$$\textit{Monitor Threshold} = \max_{u \in V_r}(m_u + 1), \tag{3.2}$$

In Equation 3.2, m_u represents the number of monitors placed on node u. Equation 3.2 increases the monitor threshold by one whenever the algorithm exceeds the current threshold number of monitors required to notice a node color change. If node color errors are less likely, we spend fewer monitors on each node, and vice versa if node errors are more common. We set the initial monitor threshold to 3 to avoid scenarios when all monitored red nodes may misreport their color in first round.

3.4 Datasets

3.4.1 Noordin Top Network

The first network studied is the Noordin Top dataset, a real terrorist network with 139 nodes and 1042 edges ('Noordin Top' is the name of the leader of this network) [38]. Its edges depict several types of relationships, including familial relationships, communications, co-attendance at meetings, etc.[5] In this network, every node is a terrorist, and we are interested in identifying those individuals that communicate using a certain medium. We consider 5 versions of this network.

[4]The optimal stopping problem studies when to take an action in order to maximize some reward. The optimal stopping problem is applicable to many disciplines including stock trading [102], oil drilling [7], and determining when to stop a random walk [78]. The solutions to the optimal stopping problem generally make assumptions about prior probability distribution of success and failure. Since we have a limited budget of monitors, calculating these priors is not feasible.

[5]Obtained from `https://sites.google.com/site/sfeverton18/research/appendix-1`.

In NoordinComs1, the POI (red nodes) are those communicating using a general computer medium; in NoordinComs2, the red nodes are those who communicate using print media; in NoordinComs3, the red nodes are those who communicate using support materials; in NoordinComs4, the red nodes are those who communicate using unknown media; in NoordinComs5, the red nodes are those who communicate using video (Table 3.3).

3.4.2 Pokec Network

The Pokec network is part of a Slovakian on-line social network.[6] The nodes in the network are users of the social network and edges depict friendship relations. Each node has some number of associated user attributes (e.g., age, region, gender, interests, height etc.). We use a sample of this network containing all nodes in the region "kosicky kraj, michalovce" and edges among them. This sampled network contains $26,220$ nodes and $241,600$ edges. Although this is not a dark network dataset, we include it for purposes of performing more comprehensive experiments and to evaluate how monitor placement algorithms perform on larger networks.

We assign node colors based on two different node attributes (Table 3.3): *age* (a node with age in the range 28-32 is marked red, and blue otherwise, giving 1736 red nodes) and *height* (a user of height less than 160 cm is marked red, giving 1668 red nodes).

3.4.3 Facebook100 Networks

Facebook100 is a collection of early Facebook networks, from when Facebook was only available to college students, and each college had its own network. [7] This

[6]Obtained from `http://snap.stanford.edu/data/`.

[7]Obtained from `https://archive.org/download/oxford-2005-facebook-matrix`.

is a collection of attributed social networks, containing node attributes such as gender, student major, year of matriculation, and dormitory of residence. In our experiments, we use select Major and Year attributes to assign node colors. Major has low attribute assortivity (homophily) whereas Year has high assortivity in these networks, and so we can evaluate the performance of monitor placement algorithms in different attribute homophily scenarios.

For each attribute, the top three most frequent (non-null) attribute values are selected to define red-blue nodes. For example, if we consider the year attribute, one set of red nodes would be all nodes that joined in year 2008 (while all others are blue). This results in six different red node and blue node assignments for each Facebook100 network. We have considered two Facebook100 college networks for our analysis resulting 6 different red-blue node assignments altogether. Table 3.3 shows selected attribute values and number of red nodes in each network.

Dataset	Nodes	Edges	Attribute	Red Nodes	No. of Red Nodes	Color Assort.
NoordinTop	139	1042	Communi.	Type1	9	0.23
				Type2	11	0.62
				Type3	9	0.11
				Type4	18	0.63
				Type5	5	0.06
Pokec	26059	241514	Age	Age=28-32	1725	0.11
			Height	Height <160cm	1663	0.06
Amherst46	2235	90954	Year	Year=2008	380	0.58
				Year=2009	377	0.85
				Year=2007	363	0.34
			Major	Major=99	200	0.04
				Major=100	174	0.03
				Major=114	161	0.03

Table 3.3: Summary of datasets and node attributes we have used to assign node colors.

3.4.4 Synthetic Terrorist Networks

To simulate dark networks embedded into civilian networks, we generate synthetic terrorist network structures corresponding to the four criminal network typologies described by [57], which we then embed into larger real social networks. We label the nodes in the criminal networks as red, and the nodes in the larger social networks as blue. Through experiments on these networks, we gain a better understanding of how the various monitor placement algorithms perform when locating individuals from different structural categories of criminal networks. Le describes these typologies in informal terms, but they can be directly mapped to concrete network structures. The network structures that we consider, and the processes that we used to generate them, are described below:

- **Standard:** Standard criminal networks follow the chain-of-command, and can be modeled using a tree structure as shown in Figure 3.2a.

  Network Generation: We generate this network as a random power-law tree with power-law exponent 3 [8].

- **Regional:** Regional criminal networks contain regional cells, each of which has a clear chain-of-command, with all regional leaders reporting to a central node. This network structure is similar to a collection of trees, where all root nodes are connected to the same central node. This network structure is shown in Figure 3.2b.

  Network Generation: We generate this network as a set of four random power law trees. We create a single central node, and connect that node to a randomly selected root node from each of the regional trees.

[8]Available at `https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.random_graphs.random_powerlaw_tree.html#networkx.generators.random_graphs.random_powerlaw_tree`

- **Clustered:** Clustered criminal networks contain multiple tightly knitted groups of criminals, with a central coordination body. An example clustered network structure is shown in Figure 3.2c.

  Network Generation: We generate this network as a set of five clusters of nodes, where each cluster is generated using a power-law cluster generating algorithm [47] [9]. A node is connected to 80% of the same cluster. We create a triangle with probability 0.8 whenever a new node is added. Nodes from non-coordinating clusters are connected to the coordinating cluster nodes randomly with 0.05 probability.

- **Core group:** Core group networks are unstructured, with a loose, flat hierarchy. An example of core group structure is shown in Figure 3.2d.

  Network Generation: We use an Erdős–Rényi random network with edge probability 0.1 to generate this structure [30] [10].

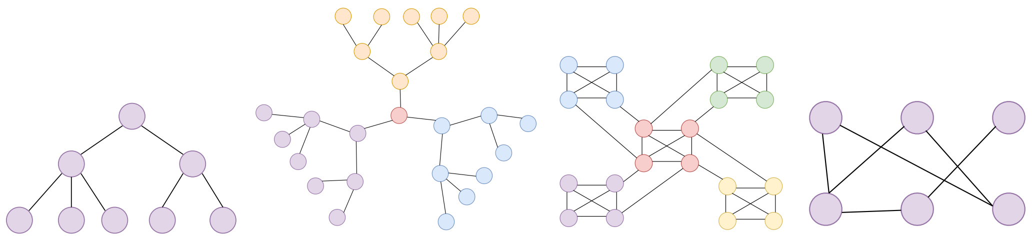


Figure 3.2: Criminal network typologies described by [57]. (a) Standard hierarchy with a chain of commands (b) Regional hierarchy with multiple chains of commands and a central controlling body (c) Clustered hierarchy with tight knit groups (d) Core group hierarchy, which doesn't exhibit clear structure

To simulate a criminal network hidden within a larger social network, we embed these synthetic networks into the Facebook100- Amherst41 network. We set

[9]Availabe at `https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.random_graphs.powerlaw_cluster_graph.html#networkx.generators.random_graphs.powerlaw_cluster_graph`

[10]Available at `https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html#networkx.generators.random_graphs.erdos_renyi_graph`

the size of the generated criminal network to be equal to 111, which is 5% of the nodes in the base social network. The number of edges in the synthetic networks vary based on the network typology, as described above. We use the network embedding process proposed by [113], which randomly connects nodes in the the criminal network to nodes in the social network. This process first selects a degree d for each node in criminal network, where d is drawn from the original degree distribution of the larger network, and then connects this node to d randomly chosen nodes from the larger social network.

3.5 Experimental Setup

In this section, we describe our experimental setup. To evaluate the performance of `RedLearn` and `RedLearnRS`, we compare them to several meaningful baseline monitor placement strategies, over a variety of network settings. In Section 3.5.1, we describe these baseline monitor placement strategies.

To perform a comprehensive evaluation of how the monitor placement algorithms perform across different error settings, we consider three categories of errors: (1) Node color errors: how monitors on red nodes report incorrect node colors, (2) Edge existence errors: how monitors on red nodes fail to report connections to other red nodes, and (3) Neighbor color errors: How node monitors may misreport neighbors' colors. These error behaviors are modeled by a variety of "error scenarios", which we describe in, respectively, Sections 3.5.3, 3.5.5, and 3.5.2. Finally, we describe our experimental setup in Section 3.5.6.

Using these experiments, we answer the following questions:

- How do the various monitor placement algorithms perform when red nodes conceal some or all of their connections to other red nodes?
- How do the various monitor placement algorithms perform when incorrect

noisy edges are present?

- How do the various monitor placement algorithms perform when the monitors misreport nodes' true color or wrongly estimate the colors of their neighbors?
- How do the various monitor placement algorithms perform on different criminal network typologies?
- Do `RedLearn` and `RedLearnRS` consistently, across a variety of misreporting scenarios, outperform the baseline monitor placement algorithms at the task of locating as many red nodes as possible?

3.5.1 Baseline Monitor Placement Algorithms

We now describe several baseline monitor placement algorithms.

In Problem Setting 1, these baseline algorithms consider all observed but unmonitored nodes to select the next node to monitor. In Problem Setting 2, when monitors on red nodes may report wrong node color, baseline algorithms consider all blue nodes for repeated monitor placement. The node with the highest score among all blue nodes and unmonitored observed nodes is selected as the next node to monitor. The same repeated monitor thresholding mechanism is used as in Section 3.3.2.

Smart Random Sampling (SR): In each step, the Smart Random Placement algorithm places a monitor on a random node, modeling chance.

Red Score (RS): The Red Score algorithm is guided by the colors reported by neighbors of a node. If a node v reports its neighbor u as red, the score associated with node u is increased by one, making it more suspicious. This algorithm selects the node with highest red score to place the next monitor. For this method, the red score is highly impacted by the accuracy of information given by the neighboring node. Additionally, due to its use of both red and blue node information, this algorithm uses the most amount of information as compared to the other baseline algorithms.

Most Red Say Red (MRSR): The MRSR algorithm places a monitor on the node with the greatest number of red neighbors who estimate it as a red node. It does not factor in blue node information and is dependent solely on the accuracy of the information given by neighboring red nodes. Blue nodes are essentially useless in this algorithm, mimicking the reality when they might not know who the POIs are. This placement algorithm would result in a red node with no red neighbors being impossible to discover except by chance.

Most Red Neighbors (MRN): The MRN placement algorithm places a monitor on the node with the most known red neighbors. As such, it is highly dependent on network's homophily. Similar to the MRSR algorithm, blue neighbors are unimportant in determining the likelihood of a given node being red.

3.5.2 Node Color Errors

In the case of monitors misreporting the neighbors' colors (representing, e.g., a guilty individual shielding himself from detection), we allow a red node to report itself as blue with some predefined probability, drawn from a normal distribution, $\mathcal{N}(0.5, 0.125)$. We conducted experiments in which blue nodes have a small probability of reporting themselves as red, and obtained similar results. For the sake of brevity, we do not include these results.

3.5.3 Edge Existence Errors

In a dark network red nodes actively try to hide their presence as well as the existence of some or all connections with other red nodes (for example, instead of using their normal cell phone to make calls to other red nodes, a red node might use a burner phone for such calls). To account for this, we consider versions of the datasets where some connections between red nodes are hidden uniformly at random. We have evaluated performance of monitor placement strategies considering how robust an algorithm is to hiding red connections. Note that this type of network presents a much more challenging setting, as one cannot simply rely on homophily to find red nodes.

3.5.4 Edge Non-Existence Errors

Any network that has been created using the data collected from node interactions and connections, is prone to noisy edges that are not necessarily a part of the actual social network. We test the performance of proposed algorithms against noisy edges by adding random edges between disconnected node pairs in the original network.

There is another type of edge non-existence errors, where criminals fabricate new edges to deceive POI detection algorithms. [73] presents how criminals can add new edges to attack network centrality measures such that leaders of criminal organizations are not the highest central nodes while keeping influence in the network. This work is not in scope for this research work.

3.5.5 Neighbor Color Errors

We assume the existence of a hierarchy among the nodes, and that nodes are more likely to conceal interactions with those above them in the hierarchy. (In the Noordin Top and synthetic terrorist networks, this hierarchy is known; in other networks, we infer it using node degree.) We assume that the red nodes are fully aware of the hierarchy, but blue nodes may or may not be aware of it, depending on the error scenario.

Consider nodes u and v, where $v \in N(u)$. The probability that a monitor on node u estimates v's color incorrectly is given by $P(e_{uv} \neq c_v)$, which depends on:

- The color of u (c_u) and color of v (c_v),
- The hierarchical position of u (L_u) relative to the position of v (L_v),
- The inherent investigation error E_u based on available information at node u.

We simulate the effort that a red node u would conceal its interactions with another criminal node v as $\frac{L_v}{L_u}$: i.e., criminals are more likely to conceal interactions with higher-ranking neighbors due to possible consequences of leaking information.

Case 1: Suppose u is a red node. The neighbor color error of a red neighbor $v \in N(u)$, is determined based on Equation 3.3.

$$P(e_{uv} \neq c_v | c_v = red) = \min\{E_u * \frac{L_v}{L_u}, 1\}. \tag{3.3}$$

Equation 3.4 defines the probability u will estimate the wrong color of a blue node. This depends on inherent estimation error, E_u

$$P(e_{uv} \neq c_v | c_v = blue) = E_u. \tag{3.4}$$

Case 2: Suppose that u is a blue node. Whether or not u is even aware of red nodes depends largely on the domain, and in particular, whether the blue nodes are part of the same organization as the red nodes (blue and red nodes are all part of the dark network, and red nodes represent a subset of interest), or if the blue nodes represent individuals who are not part of the same organization as the red nodes (e.g., the red nodes represent dark nodes in a sea of blue node civilians).

- Neighbor color error type 1 (All nodes aware of red nodes).

  Here, $P(e_{uv} \neq c_v | c_u = blue, c_v = red)$ is determined using Equation 3.3, since blue nodes know about red nodes and their hierarchy. Additionally, monitors which are placed on blue nodes may misreport colors of blue nodes depending on inherent analysts' error as mentioned in Equation 3.4.

- Neighbor color error type 2 (Only red nodes aware of other red nodes).

  In this case, it is not possible to distinguish whether a neighbor is red/blue by looking at their emails, messages, etc., because blue nodes don't know that their red neighbors are red. Here, blue nodes will simply report that all their neighbors are blue. Because of this $P(e_{uv} \neq c_v | c_u = blue, c_v = blue) = 0$ and $P(e_{uv} \neq c_v | u = blue, v = red) = 1$.

In all cases, if $P(u\ lie\ v)$ is greater than 1, it is rounded down to 1.

3.5.6 Experimental Settings

Because our error types are probabilistic, and the inherent neighbor color error E_u of each node u may also change, for each network and lying scenario, we perform 25 runs of each monitor placement algorithm. For a fair comparison across different monitor placement strategies, we run each monitor placement algorithm with the same settings (including the same red starting node).

Monitor Budget: In each run, we consider budgets of up to half the number of nodes in the network. We do this to illustrate the behavior of the algorithm over both small and large budgets, even though in practice it is unlikely that fully half of the nodes can be queried.

Training Learning Algorithms: The Noordin Top network is small, and so we retrain `RedLearn` and `RedLearnRS` after each monitor is placed. The Pokec, Facebook, and Synthetic networks are larger, so for the sake of efficiency, we train the learning model once per every 20 monitors placed.

Node color errors: A monitored red node misreports its color with some probability drawn from a normal distribution, $\mathcal{N}(0.5, 0.125)$. We consider this to be an error in the node color determination mechanism. Therefore, all monitored red nodes misreport their colors with the same probability as described in Section 3.5.2.

Edge existence errors: In a given network, we consider that only red nodes try to conceal their connections with other red nodes with a fixed probability. We present results when red nodes hide their connections with probability 0 (original network), 0.5 (hiding half of red edges) and 1.0 (hide all red connections).

Edge non-existence errors: We evaluate monitor placement algorithm performance in the presence of noise (incorrect edges) by adding edges with a probability of 0.01, between any two disconnected nodes in the original network.

Neighbor color errors: In these experiments, the inherent neighbor color error at each node is drawn from a normal distribution, $E \sim \mathcal{N}(0.5, 0.125)$. For the Noordin Top network, ground truth hierarchy scores are known, as shown in Table 3.4. In the Pokec and Facebook 100 networks, we set the hierarchy score to be the degree of the node.[11] For the synthetic networks, hierarchy assignments are as follows: in the standard and regional typologies, the hierarchy score of a node is equal to the number of descendant nodes that it has; and in the other typologies, the hierarchy score of a node is equal to its degree. Given some error type, a monitored node u, misreports neighbor v's color with probability $P(e_{uv} \neq c_v)$ as mentioned in Section 3.5.5.

[11]Results were similar for different types of centrality, including eigenvector and betweeneness centrality.

Role	Hierarchy score	No. of nodes
Strategist	5	10
Commander; Religious Leader	4	23
Trainer/instructor; Bomb maker; Facilitator; Propagandist; Recruiter	3	33
Bomber/fighter; Suicide Bomber; Courier; Recon/Surveillance	2	33
Unknown	1	40

Table 3.4: Noordin Top network hierarchy assignment

3.6 Results and Analysis

This section is organized as follows: Because node classification algorithms appear to be a natural approach to this problem, we begin by presenting a comparison of `RedLearn` against the ICA node classification algorithm in Section 3.6.1. Next, we consider Problem Setting 1, in which monitors accurately report the color of the occupied node. We compare `RedLearn` to the baseline monitor strategies discussed in Section 3.5.1. We examine how the performances of these monitor placement algorithms are affected by (1) red nodes concealing their connections with other red nodes with probability 0, 0.5 and 1.0, (2) the neighbor color error type used by the nodes, and (3) the monitor placement budget.

We next move to Problem Setting 2, where monitors may misreport the color of the occupied node. In Section 3.6.3, we evaluate the performance of `RedLearnRS` as compared to baseline algorithms.

In Section 3.6.4, we evaluate performance of monitor placement algorithms when nodes report incorrect edges. We then conduct a feature importance analysis in Section 3.6.5 to determine the most important features for `RedLearn` and `RedLearnRS` under different neighbor color error types and edge existence errors. Finally in Section 3.6.6, we evaluate `RedLearnRS` on the various dark network typologies discussed in Section 3.4.4

3.6.1 Node Classification Algorithms

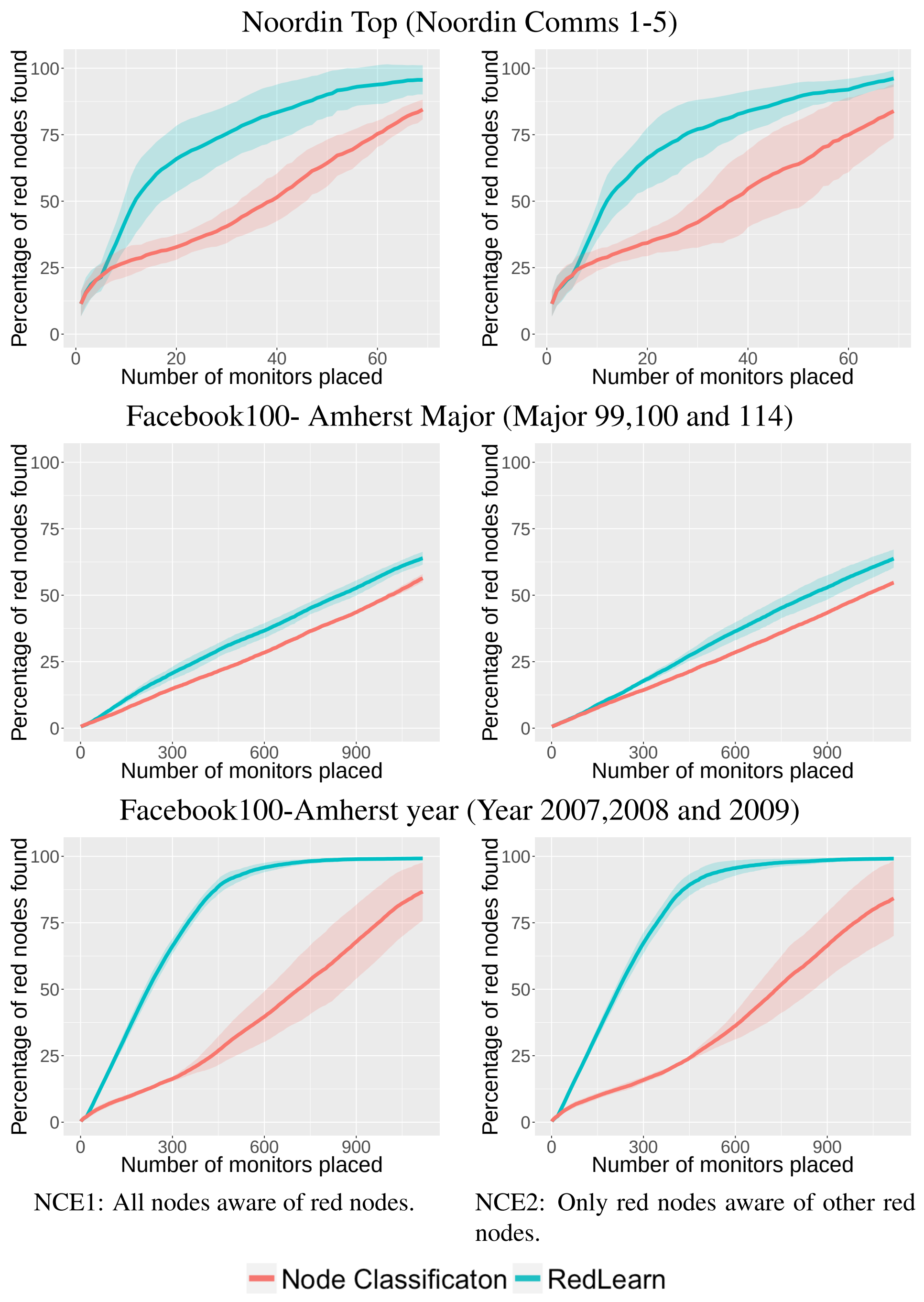


Figure 3.3: Comparison of average performance of ICA and `RedLearn` on the NoordinTop terrorist network and Facebook-100 networks. RedLearn outperforms ICA in all networks for both neighbor color error types considered.

In this section, we compare performance of the node classification algorithm (ICA) to `RedLearn`. Figure 3.3 compares the performance of ICA to `RedLearn` on several networks, under Problem Setting 1, for the two neighbor color error settings. In both algorithms, we used logistic regression as the base classifier with the same set of features. `RedLearn` outperforms ICA in both error types in all networks we have considered. A similar trend is followed even when we introduce edge existence errors by concealing edges among red nodes.

ICA performs poorly because the training is performed on a sampled network, and complete information about network topology is necessary to provide accurate predictions. Prediction errors are propagated throughout the predictions in ICA, because ICA uses predicted labels to calculate features. Even though ICA performs well in node classification problems when complete information is present, it exhibits poor performance in partial informaiton scenario.

On the other hand, `RedLearn` uses both node colors that are reported by the monitors and connections in making accurate predictions about node colors. `RedLearn` does not give priority to network's topology while making the predictions, whereas ICA heavily depends on topology to calculate neighborhood based features.

3.6.2 Problem Setting 1: Node color reliable

In this section we present results for the monitor placement algorithms for Problem Setting 1. Figure 3.4 shows the performance of `RedLearn` on the NoordinComs4 network for (a) when no edge existence errors are present, (b) red edges are concealed with probability 0.5, and (c) all red edges are concealed. When all edges between red nodes are available, the problem becomes easy, and the simple algorithm of monitoring the node with the most red neighbors (MRN) is best (because the red nodes exist in a near-clique). However, note that in both lying scenarios, even when all red edges are revealed, `RedLearn` is second to MRN.

However, we see that as edges between red nodes are increasingly concealed, the MRN performance worsens. In such cases, `RedLearn` performs much better

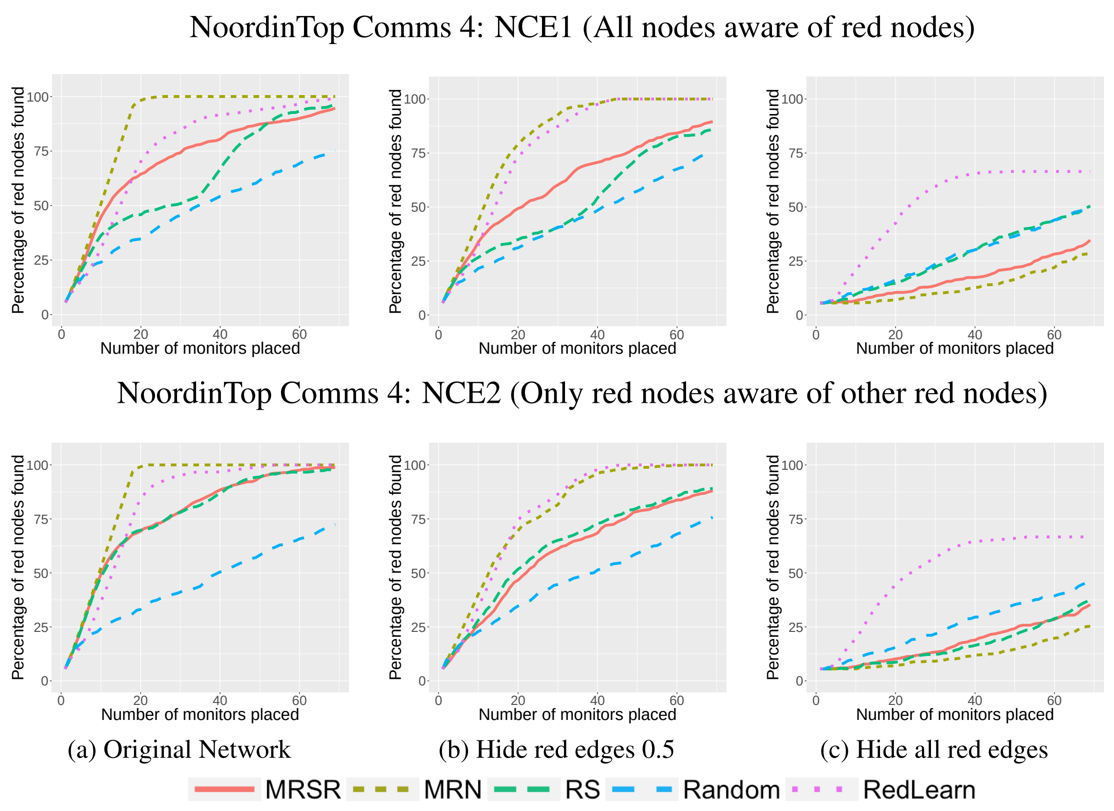


Figure 3.4: Comparison of monitor placement algorithms on the NoordinComs4 network when reported node colors are reliable (Problem Setting 1). MostRedNeighbors (MRN) performs well when there are no edge existence errors. `RedLearn` performs consistently well across all edge existence error scenarios.

than all comparison methods: it is able to learn the patterns of reporting and structural characteristics of red nodes, achieving the highest performance.

We present the average performance of monitor placement algorithms on different attributed social networks in Figures 3.5 and 3.6. Figure 3.5 shows monitor placement algorithm performances for neighbor color error type 1, and Figure 3.6 shows average performance of neighbor color error type 2.

The NoordinTop and Facebook100-Year networks have very high node color homophily, and so all monitor placement algorithms perform well compared to random placement. However, in Facebook100-Major and Pokec networks, node color homophily is very low, and we observe overall lower performance of the various strategies. MRN and `RedLearn` perform equally well when there are no edge existence errors, and all red edges are correctly reported, but as before, when red edges are concealed, `RedLearn` is generally the best.

We see similar patterns across all networks: edges between red nodes help select the node with the most red neighbors and MRN outperforms all algorithms closely followed by `RedLearn`; but when these edges are concealed, `RedLearn` is the clear winner.

3.6.3 Problem Setting 2: Node colors misreported

In this section we evaluate the performance of monitor placement algorithms, including `RedLearnRS`, when node colors are misreported by monitors.

Figure 3.7 shows that the results on the NoordinComs4 network follow a similar pattern as in Problem Setting 1: when reported edges are reliable, MRN performs well, with `RedLearnRS` close behind; as we introduce edge existence errors by hiding edges among red nodes, `RedLearnRS` becomes the clear winner.

All monitor placement strategies are affected by node color errors, because they must expend greater amounts of budget to repeatedly monitor the same node. Thus, the performance of the baseline algorithms is reduced in this problem setting, since they do not employ a mechanism to find possible node color inaccuracies. In placing the next monitor, they have to consider all believed-blue nodes and unmonitored nodes. In contrast, `RedLearnRS` identifies possible node color errors among blue nodes using the outlier detection algorithm, and only considers these error nodes and unmonitored nodes when placing the next monitor. `RedLearnRS` performs similarly across the different neighbor color error types considered.

We summarize the average performance of monitor placement algorithms across all networks in Figures 3.8 (NCE1) and 3.9 (NCE2). In all networks where attribute assortivity (homophily) is low (Pokec, Facebook100-Major networks), `RedLearnRS` out performs all other monitor placement strategies. When attributes show ho-

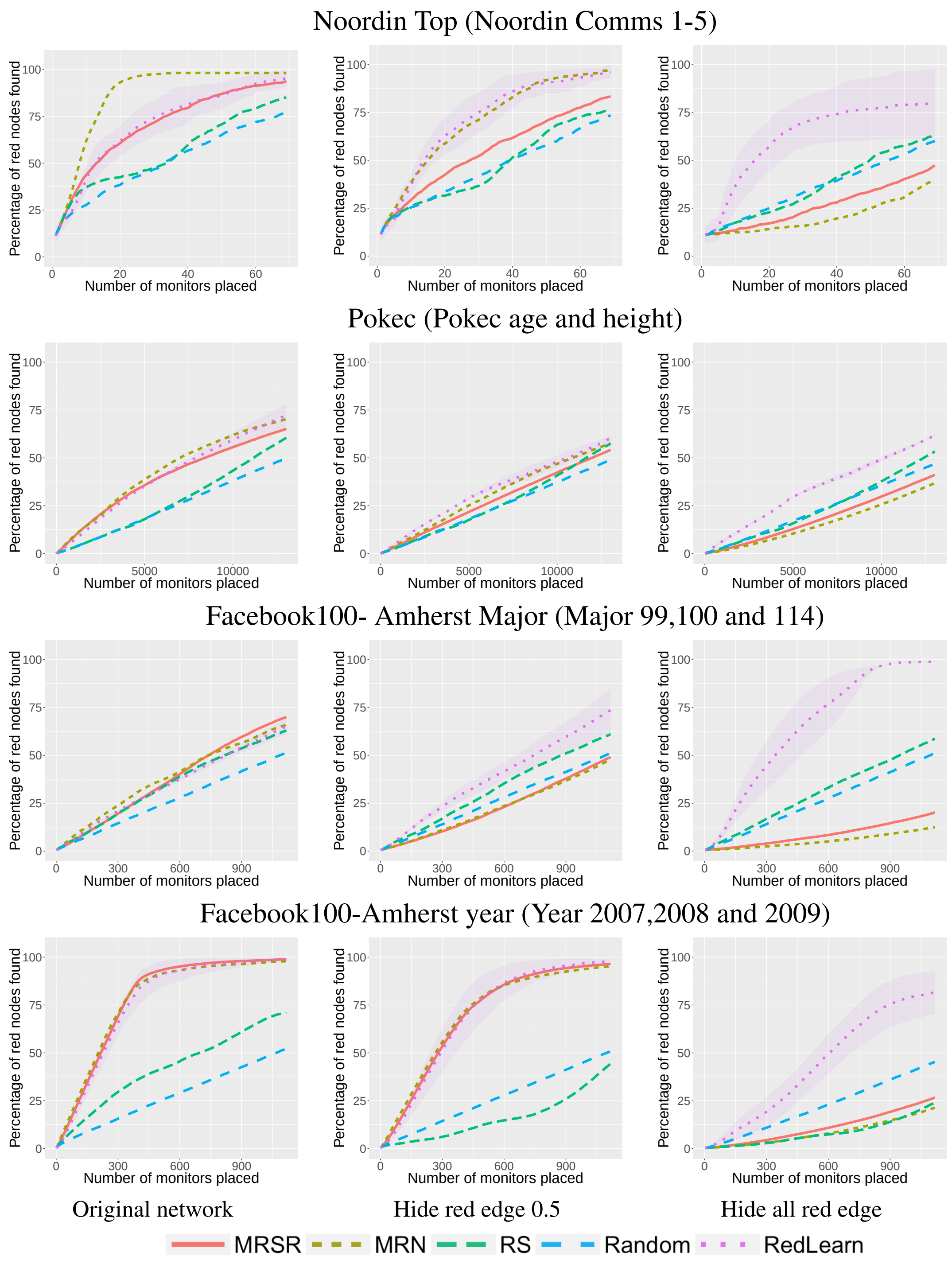


Figure 3.5: Monitor placement algorithm average performance when reported node color is reliable (Problem Setting 1). Neighbor color error type 1 (NCE1:All nodes aware of red nodes) is considered here. MostRedNeighbors (MRN) and `RedLearn` algorithms perform similarly when there are no edge existence errors. As we introduce these errors,`RedLearn` becomes the clear winner across all monitor placement algorithms.

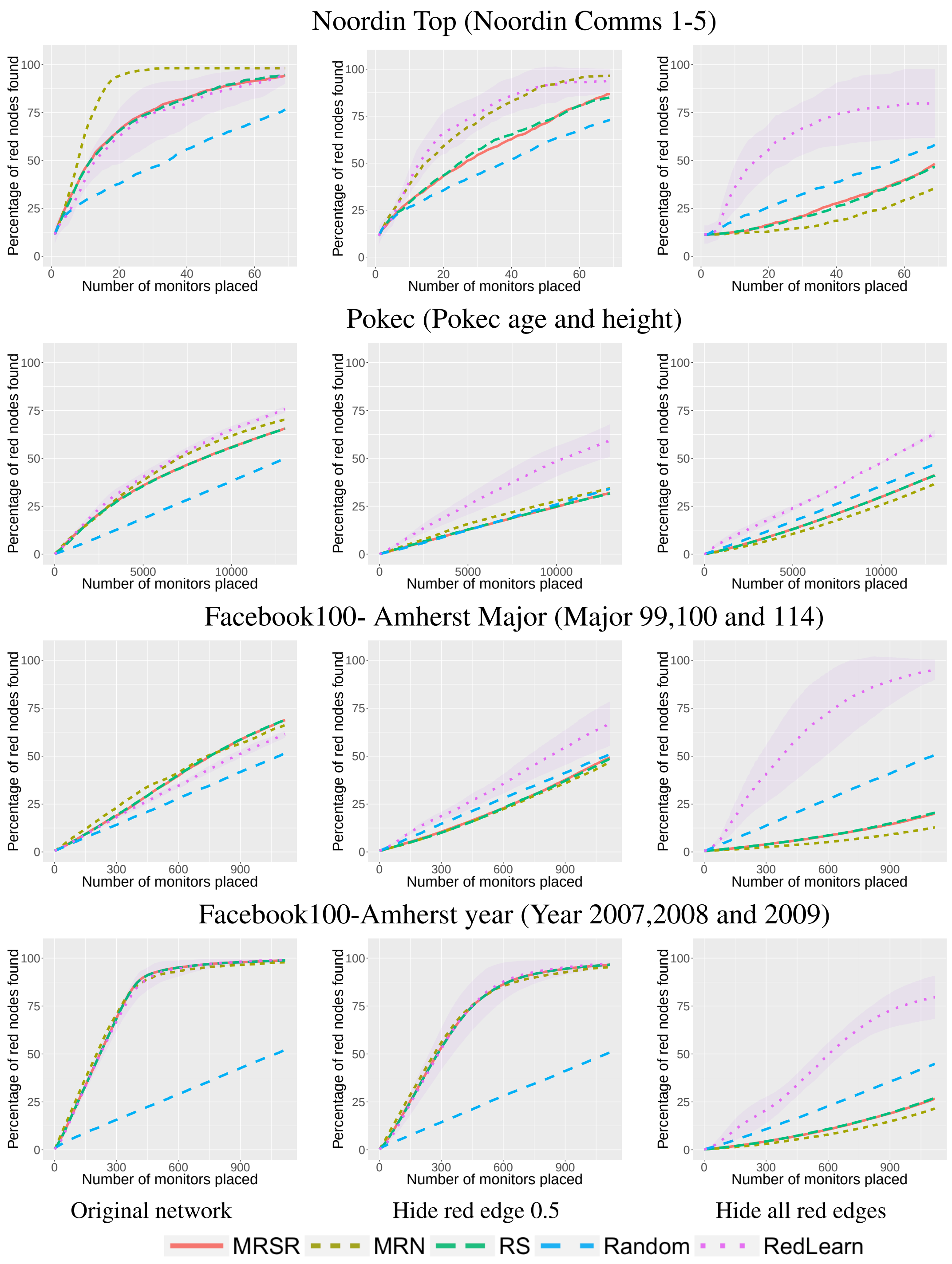


Figure 3.6: Monitor placement algorithm average performance when reported node colors are reliable (Problem Setting 1). Neighbor color error type 2 (NCE2: Only red nodes aware of other red nodes) is considered here. `RedLearn` has matched the performance of MostRedNeighbors (MRN), in most networks when there are no edge existence errors. As red nodes hide their connctions with other red nodes, `RedLearn` becomes the clear winner.

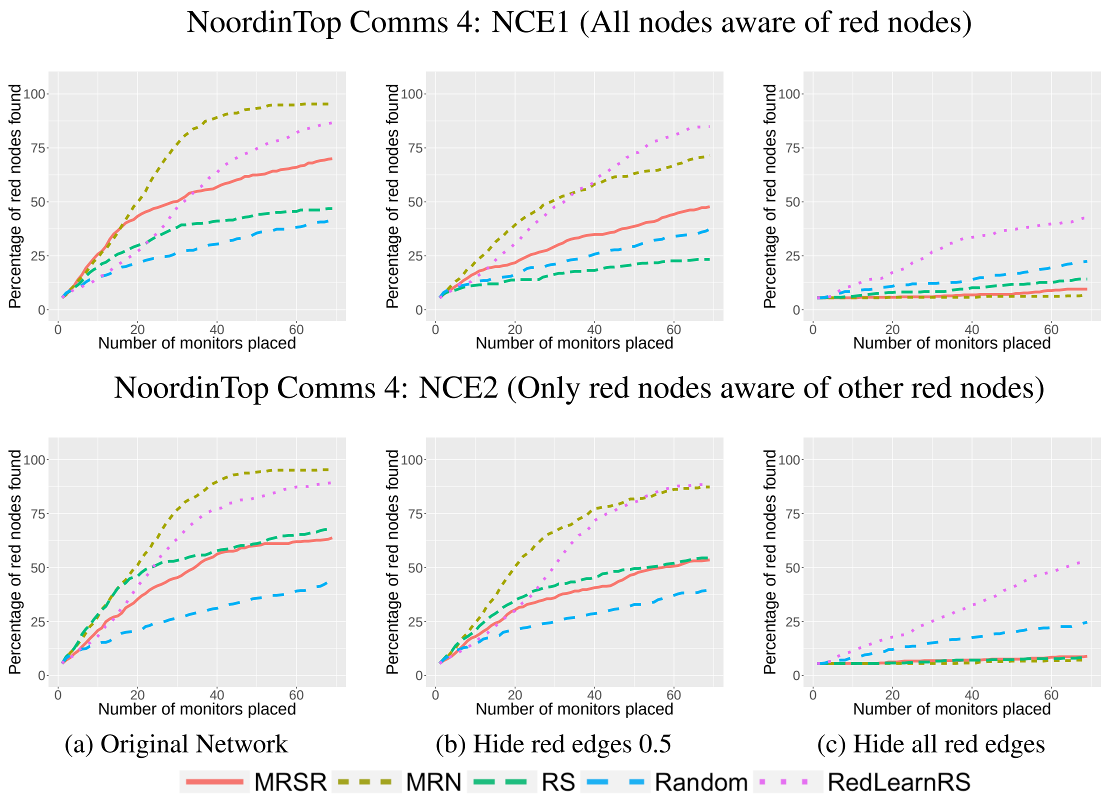


Figure 3.7: Comparison of monitor placement algorithms on the NoordinComs4 network when monitors misreport node colors (Problem Setting 2). Even though MostRedNeighbors (MRN) performs well in the original network, its performance worsens when edge existence errors are introduced. `RedLearn` performs consistently well across all neighbor color error types and edge existence error scenarios.

mophily, `RedLearnRS` performs quite similar to the best monitor placement algorithm (MRN).

Our conclusion is that as we conceal red edges, `RedLearnRS` is the best performing monitor placement algorithm. While it comes in second in a few cases in which all the original edges between red nodes are known, it closely follows the performance of MRN, and it outperforms it in some cases. Also, the situation of all the existing edges being known is not realistic, so the performance of `RedLearnRS` in the cases when some or all of these edges are missing is most relevant to dark networks.

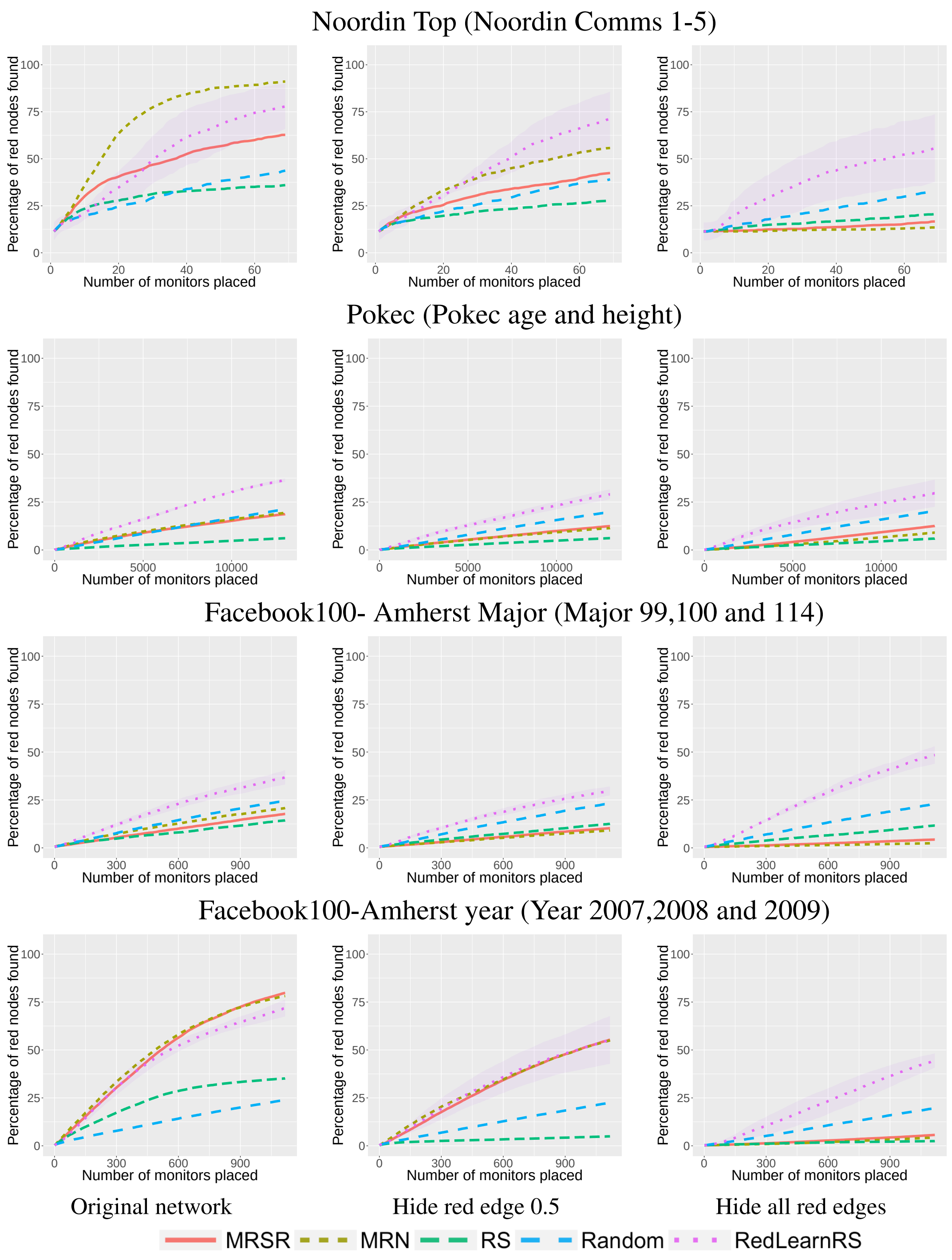


Figure 3.8: Monitor placement algorithm average performance when node colors are misreported by monitors (Problem Setting 2), under neighbor color error type 1 (NCE1: all nodes are aware of red nodes). MostRedNeighbors (MRN) performs well when there are no edge existence errors, but worsens as we introduce such errors. `RedLearn` performs consistently well across all edge existence error scenarios.

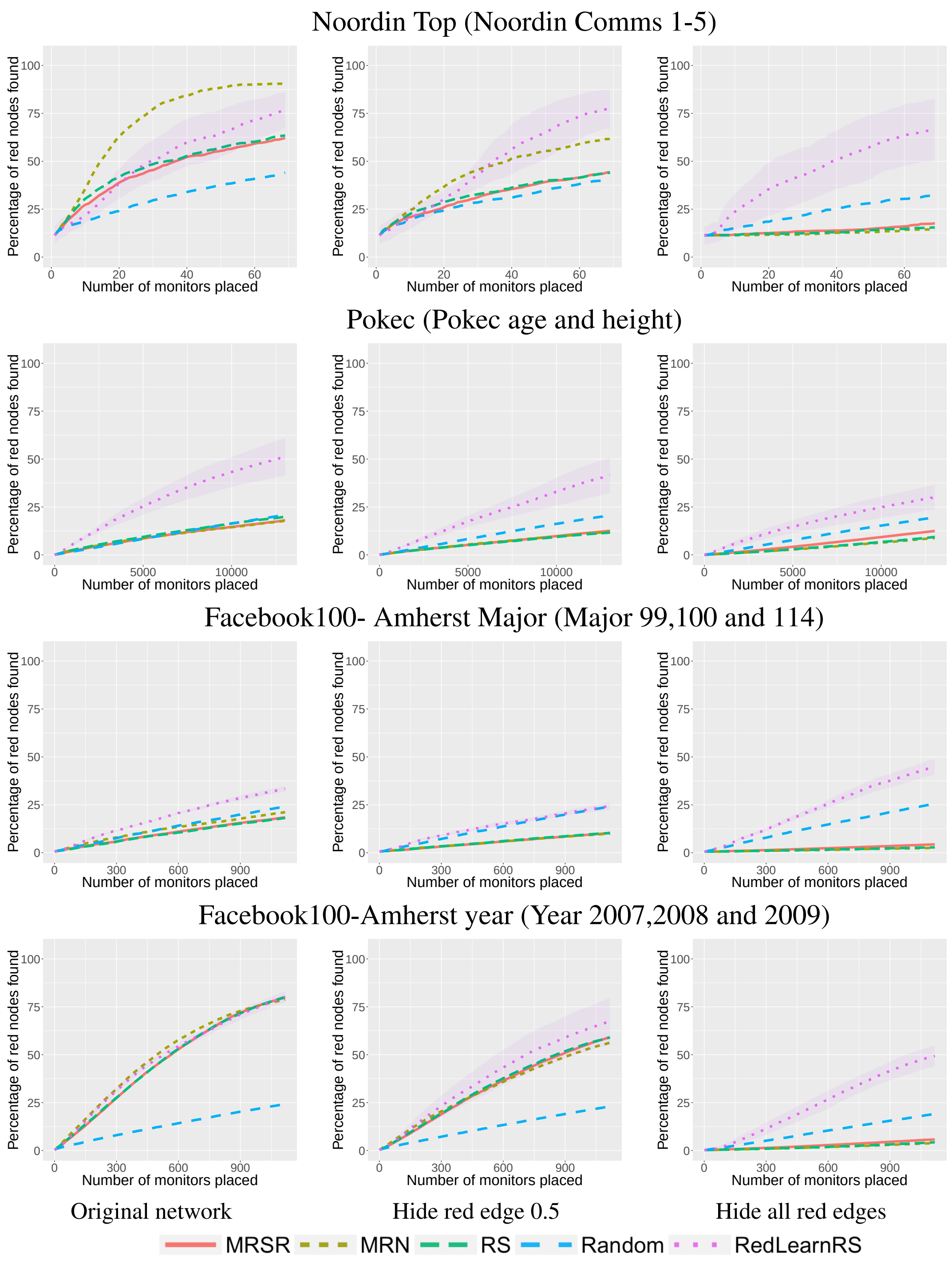


Figure 3.9: Average performance of monitor placement algorithms when node colors are misreported (Problem Setting 2), under neighbor color error type 2 (NCE2: Only red nodes are aware of other red nodes). `RedLearn` performs consistently well across all edge existence error scenarios, when other monitor placement algorithms are affected heavily by concealing red edges.

3.6.4 Edge non-existence error analysis

In this section we evaluate performance of monitor placement algorithms when the monitors report incorrect edges. This case is the most realistic one, incorporating noise that is normally present in real data collection. We included random edges between disconnected nodes in the original network with probability 0.01 to evaluate this phenomenon.

Figures 3.10 and 3.11 present average results for NoordinTop networks. The performance of all monitor placement algorithms is not drastically affected by the incorrect edges that were added to the network.

]

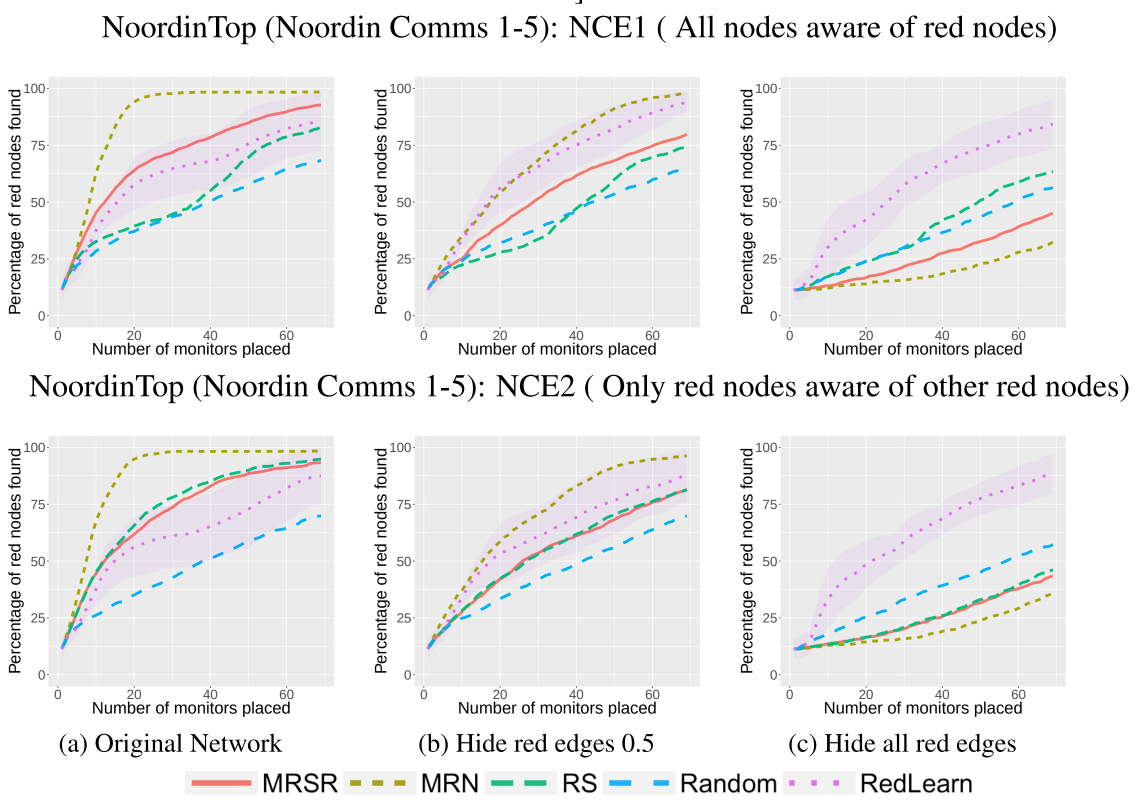
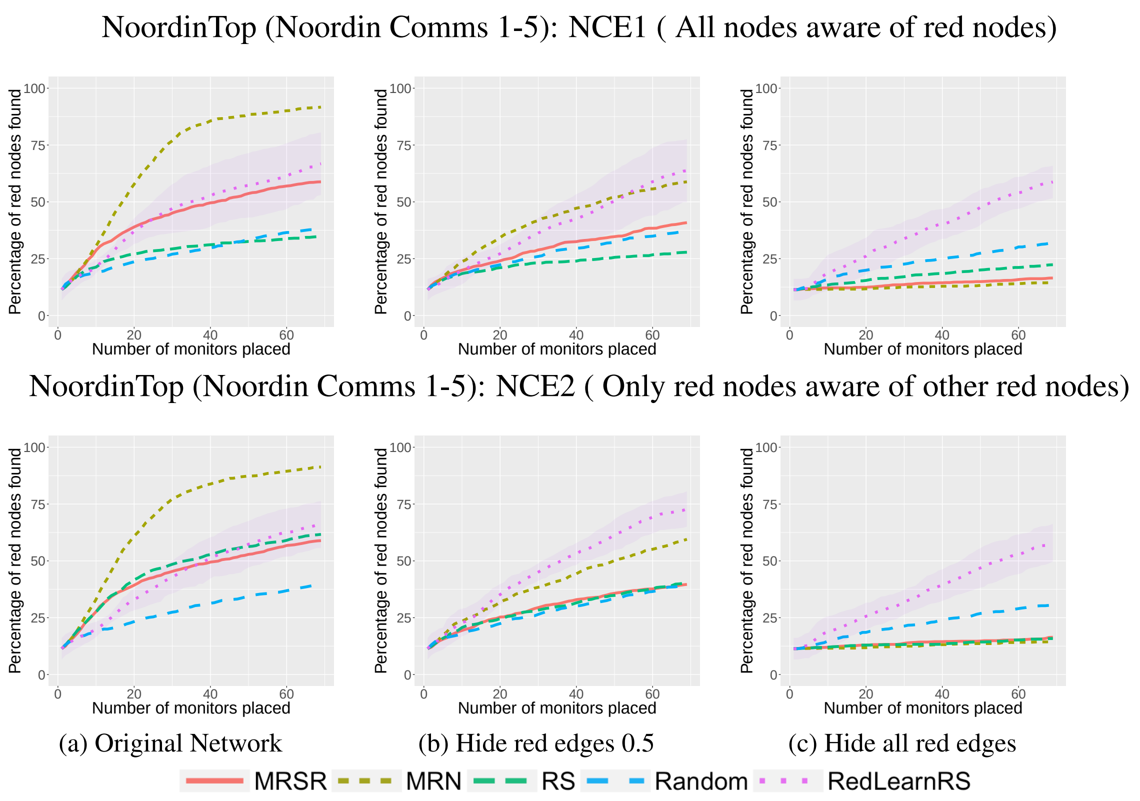


Figure 3.10: Average performance of monitor placement algorithms in NoordinTop networks when monitors report incorrect edges. Problem Setting 1: Node colors are reliable is considered here. Performance of monitor placement algorithms is not heavily affected by noisy edges.

We observe a similar pattern of results for all networks and all error scenarios



(a) Original Network (b) Hide red edges 0.5 (c) Hide all red edges

Figure 3.11: Average performance of monitor placement algorithms in NoordinTop networks when monitors report incorrect edge. Problem Setting 2: Node colors are misreported is considered here. Performance of monitor placement algorithms remains quite similar to when there are no noisy edges.

we have considered in this chapter.

3.6.5 Feature Importance Analysis

In this section, we determine the most important features for good performance for `RedLearn` and `RedLearnRS`. We look at how feature importance changes when red nodes do or do not show homophily, as well as when node colors are misreported. We examine the NoordinTop terrorist networks and Facebook100 Amherst-Major networks for our analysis, since these networks show high and low color assortivity respectively.

We have divided features in to four different categories to better explain feature importance: (1) Neighbor based: Number of blue/red neighbors (2) Tight knit groups: Number of red triangles, (3) Neighbor estimates based: Number red/blue neighbors estimate red/blue, red score and inferred probability of red, and (4) Second hop neighbors: Second hop red/blue neighbors. When node colors are misreported (Problem Setting 2), we consider an additional repeated monitor category, the number of monitor placed that blue/red nodes lied about.

We evaluate feature importance using absolute values of the coefficients of each feature at the logistic regression decision boundary. The average feature importance of each category is reported.

Problem Setting 1: Node color reliable

Figure 3.12 presents average feature importance scores for NoordinTop terrorist networks and Facebook100 Amherst-Major networks. Tight knit groups are the most important feature group for `RedLearn` in NoordinTop networks. This agrees with our intuition, as in NoordinTop networks red nodes are mutually adjancent. Neighbor estimates do not have a high importance in NoordinTop networks, since red nodes can be easily identified using neighborhood features. In Facebook100 Amherst-Major networks neighbor color estimates, neighbors and tight knit groups are equally important since red nodes are much difficult to distinguish from blue nodes.

Feature importance changes as we introduce different types of errors into the problem. When we introduce edge-existence type of errors (by hiding edges between red nodes), the neighbor feature category improves importance in both network types. This is primarily due to `RedLearn` learning that red nodes tend to have fewer red neighbors, and it obtains outstanding performance when we remove edges among red nodes. Other neighborhood based monitor placement algorithms are not capable of learning this phenomenon. Feature importance scores remain quite similar across different neighbor color error types considered.

Problem Setting 2: Node color misreported

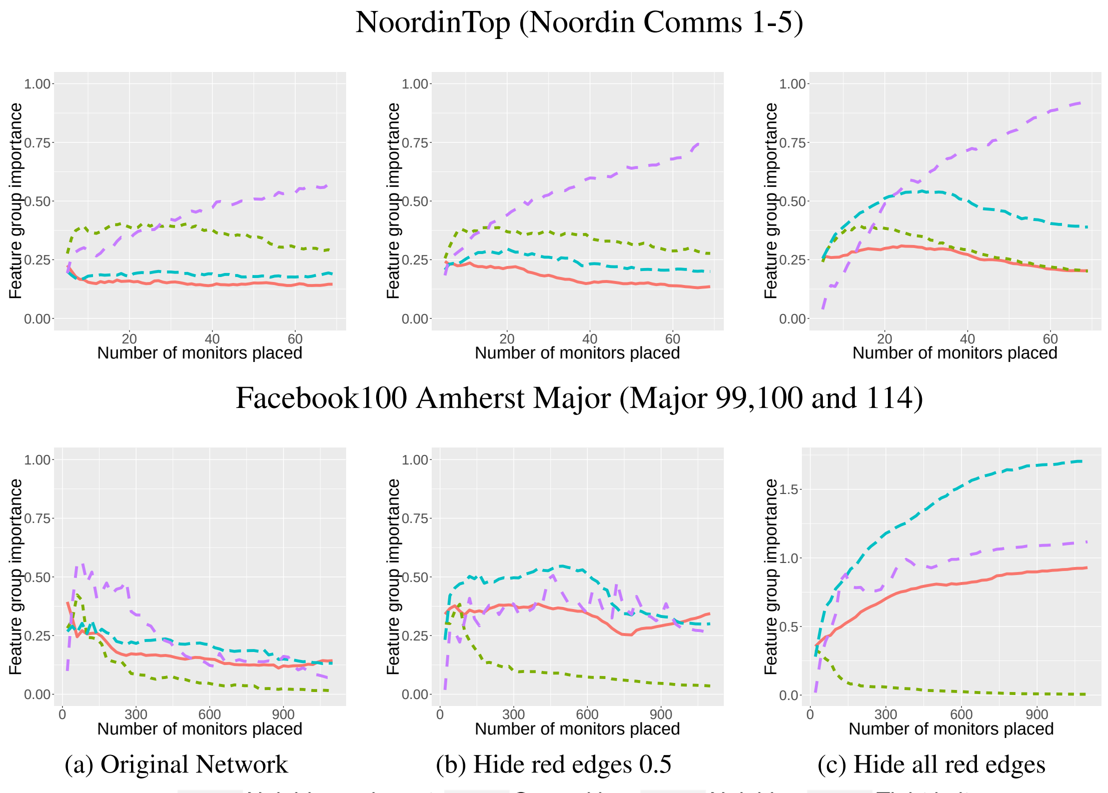


Figure 3.12: Average feature importance scores of NoordinTop and Facebook100 Amherst-Major networks when reported node colors are reliable (Problem Setting 1). Tight knit groups are the most important feature category in NoordinTop networks, since red nodes tend to be clustered together. The importance of the neighbor feature category increases when we hide red edges.

Figure 3.13 shows average feature importance scores for `RedLearnRS`. We add the Repeated Monitor feature category, which was introduced to determine whether to place a monitor on an already monitored node. This feature category has high importance in both networks considered. Other feature categories have similar importance scores as Problem Setting 1 as seen in Figure 3.12.

3.6.6 Network structure analysis

In this section, we discuss how monitor placement algorithms perform for several real network structures observed in terrorist networks. Recall that we create these networks by embedding different synthetically created terrorist network hierar-

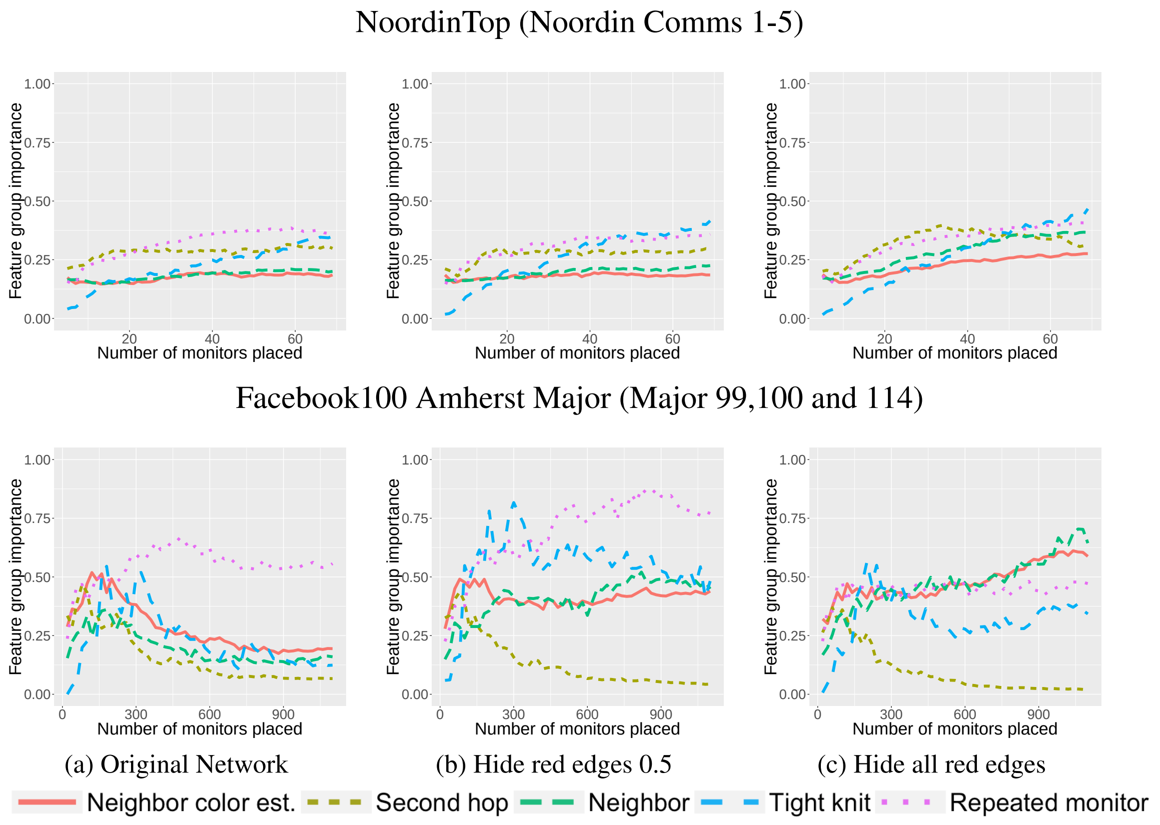


Figure 3.13: Average feature importance scores of NoordinTop and Facebook100 Amherst-Major networks when node colors may be misreported by monitors. Repeated monitor feature group has the highest importance in most networks since this feature category determines whether to place a monitor on an already monitored node.

chies into the Facebook100 Amhrest social network, as discussed in Section 3.4.4.

Figure 3.14 depicts the performance of monitor placement algorithms when node colors reported by monitors are reliable (Problem Setting 1). We see that in standard and regional hierarchies, `RedLearn` outperforms the other monitor placement strategies, and the MRN method performs poorly. This occurs because standard and regional hierarchies have a chain of command and follows a tree structure, so each node will be connected to very limited number of red nodes. In such a structure, `RedLearn` is able learn the neighborhood characteristics of a red node.

In neighbor color error type 2, the performance is affected by only red nodes' knowing colors of other red nodes. Because each red node is connected to a very

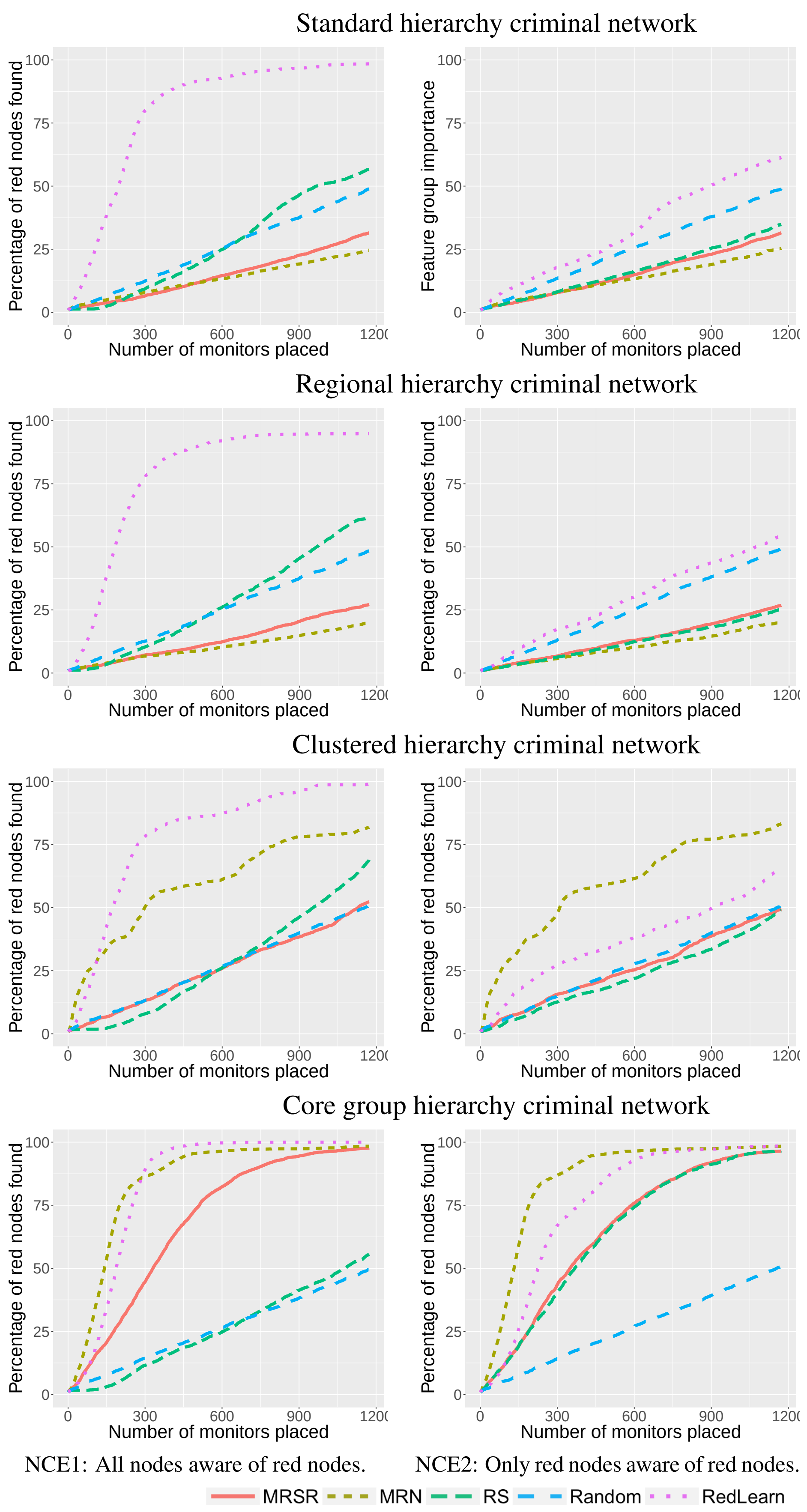


Figure 3.14: Comparison of monitor placement algorithm performance for different criminal network hierarchies when reported node colors are reliable (Problem Setting 1). `RedLearn` performs the best in all network hierarchies in NCE 1. When red nodes are clustered and/or connected to lot of other red nodes (core group), MRN performs well.

limited number of other red nodes, neighbor color estimates do not reveal much information.

In clustered and core group hierarchies, most red neighbors (MRN) performs well because red nodes are connected to many other red nodes. Overall, RedLearn performs consistently well across all types of terrorist network structures.

Figure 3.15 shows monitor placement algorithm performance when the colors of monitored nodes may be misreported (Problem Setting 2). In this setting, RedLearnRS outperforms all monitor placement strategies in all network hierarchies and neighbor color error types. In the clustered hierarchy and core group hierarchy, MRN performs quite well, but it is not able to beat RedLearn. Overall, the baseline algorithms perform worse than random monitor placement in most network hierarchies, because they cannot identify blue nodes with color errors.

3.7 Discussion and Conclusion

Members of dark networks conceal information by nature, and while these networks are deceptive and sparse, they are still structured. Based on these properties, we created and analyzed the results of several methods of sampling the networks to identify People of Interest (POI, or 'red nodes'). We tested these methods on a small real terrorist network, larger social networks, and synthetic terrorist networks embedded into real social networks. As our monitors represent analysts that may misreport findings, we used several error scenario models, including node color errors, neighbor color errors, neighbor existence error and neighbor non-existence errors.

We created RedLearn, a learning-based method for locating POI in dark networks when reported node colors are reliable (Problem Setting 1). REDLEARN uses features from the simpler baseline algorithms and learns how to identify red

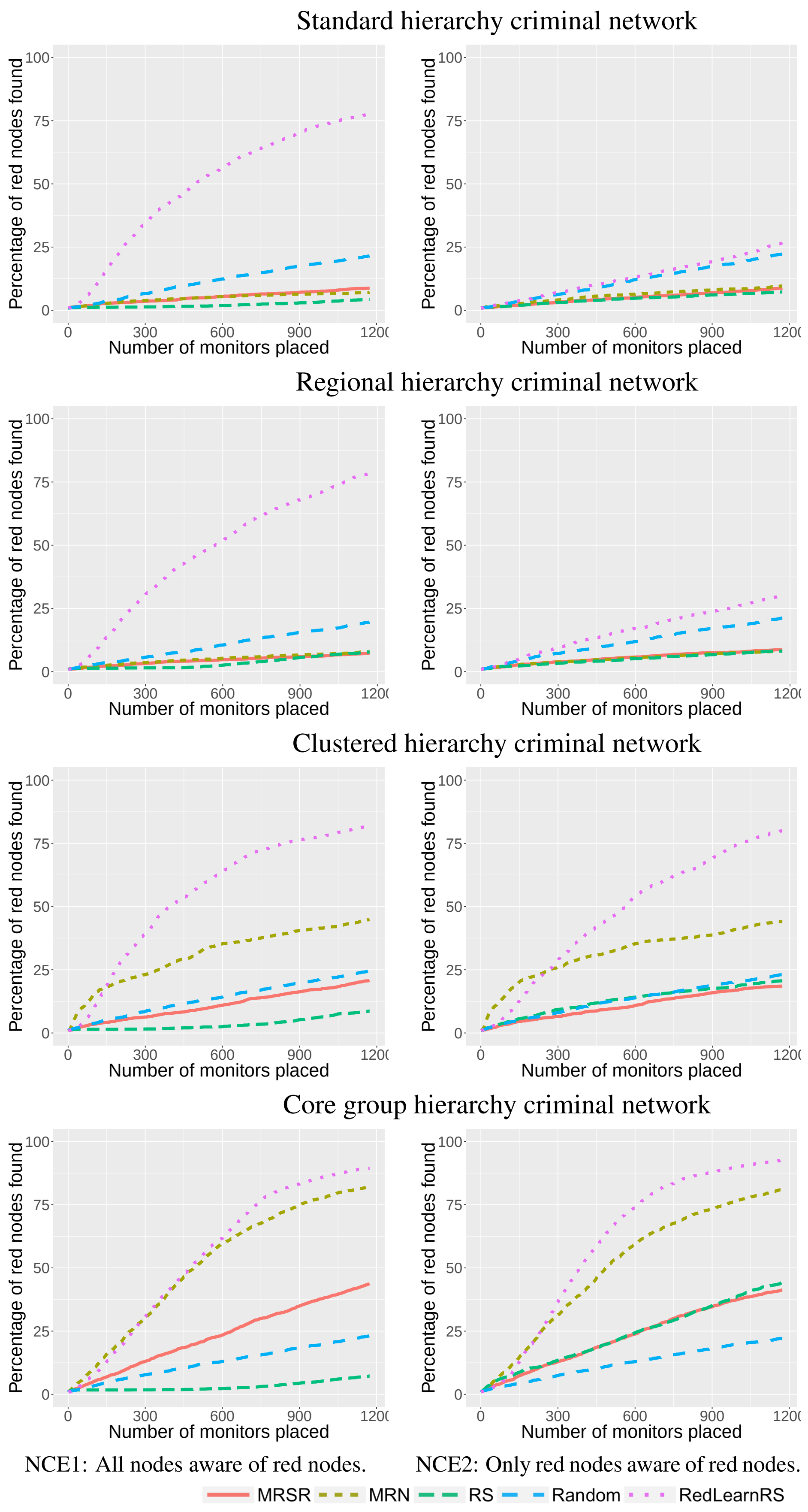


Figure 3.15: Comparison of monitor placement algorithm performance for different criminal network hierarchies when node colors may be misrported by monitors (Problem Setting 2). `RedLearnRS` outperforms all other monitor placement strategies.

nodes in networks. Results show that `RedLearn` outperforms the other methods in cases where one cannot rely on nodes to reveal all their connections (edge existence errors). `RedLearn` performed well across the different neighbor color error types we considered.

We then introduced Problem Setting 2, where node colors may be reported with errors. In this setting, we used `RedLearnRS`, a re-sampling algorihtm that extends `RedLearn` to identify POIs. All monitor placement algorithm performance were affected in this case since multiple monitors needed to be placed on a single node to uncover the true color, however `RedLearnRS` outperformed other baseline monitor placement algorithms by successfully identifying POIs that were initially misreported.

`RedLearnRS` assumes that only red nodes will misreport their color. Therefore it is necessary to re-monitor nodes that are reported to be blue to identify potential node color errors. This might lead to the risk of repeatedly monitoring innocent blue nodes. We introduced repeated monitor threshold in `RedLearnRS` to minimize number of times a blue node will be monitored. Our analysis shows that the `RedLearnRS` algorithm re-samples a blue node 1.61 times on average with a standard deviation of 1.2 compared to 1.68 average and 1.03 standard deviation of re-sampling a red node. This shows that `RedLearnRS` does not excessively monitor the same blue node. Another important fact is that when we refer to "monitoring" some node, it doesn't necessarily mean that the node will be arrested and questioned. It is rather an analyst will spend more time and resources determining the true color of a node.

In general, all monitor placement algorithms performed well in networks where red nodes tend to have direct connections with other red nodes (high color assortivity present in Noordin Top, Facebook100-Year attribute); and vice versa when red node assortivity is low (Facebook100-Major attribute and Pokec networks).

The MostRedNeighbors (MRN) monitor placement algorithm performed the best in highly color assortative networks since red nodes tend to have many red neighbors. The performance of MRN was drastically affected by edge existence errors . On the other hand, `RedLearn` performed well across all network types and error types. All monitor placement algorithms are not affected by small edge non-existence errors

`RedLearn` and `RedLearnRS` use several network structure based features and neighbor color estimate based features to identify nodes of interest in social networks. We conducted a feature importance analysis to determine most important features in different network structures and problem settings. Tight knit groups were seen to be most important feature group when red node assortivity is high (NoordinTop networks). For other networks, Neighbors, Neighbor color estimates and Tight knit groups showed to be equally important. Repeated monitors feature category was the most important feature category in Problem Setting 2, when node colors are reported with errors.

We generated several criminal network hierarchies in accordance with the criminology literature, and embedded these synthetic criminal network structures into Facebook100-Amherst network to determine which algorithms perform well across different criminal hierarchies. Our results showed that MRN monitor placement works well when criminals are clustered together in groups. In all other hierarchies `RedLearn` and `RedLearnRS` outperform baseline monitor placement algorithms across two problem settings and error types we have considered.

The proposed algorithms only consider limited information to predict which individual should be monitored next. Monitoring an individual can lead to violating someone's privacy or can have an adverse effect on the individual. Therefore, when using the proposed algorithms, careful attention should be made to avoid algorithmic biases (such as targeting some specific race since an identified individual belongs to that race). Further, human intervention is essential to further evaluate the predictions made by the algorithms to limit unnecessary monitoring of civilians. Using the proposed algorithms as a guide to limit the search space would be the best way to incorporate the algorithm in an actual criminal investigation.

One interesting future direction would be to consider this problem from a criminal's perspective to analyze how criminals can deceive these learning algorithms. This analysis can then be used to develop algorithms that are robust to deceptive criminal behavior.

CHAPTER 4 MODELING PEOPLE OF INTEREST IN SOCIAL NETWORKS

Social network analysis is often used to gain fundamental insights into human social interactions. For example, one can identify the most influential individuals in a network, understand how information spreads through groups, and characterize the community structure of a population. Typically, when one performs such analysis, one assumes the availability of accurate information about the network structure and people who are part of it.

However, in certain applications, such assumptions about data accuracy may not hold. For example, in Chapter 3, we proposed algorithms to sample POIs in dark networks where there is a covert group (such as a criminal gang or terrorist cell) hiding among civilians in a social network. The nodes representing these people are adversarial against those who seek to collect or analyze their data. Thus, the covert nodes may misreport data about their network structure or attributes (their own or others') to a data collector in order to hinder such analysis and hide the identities of themselves and their compatriots. While most data analysis tasks must deal with noisy data, this type of adversarial behavior seeks to deliberately deceive an analyst, and may present different challenges than simply random noise.

In Chapter 3, we discussed various types of errors covert entities can introduce to hinder the adversarial network data collection. To better understand adversarial network structure, one should be able to characterize *how* individuals in these networks may attempt to deceive a data collector. Such information can be used to design network analysis algorithms that are more robust against purposeful misinformation, characterize the network structures that promote adversarial behaviors, identify regions of the network for which more accurate data should be collected (by, e.g., recruiting informants from the group), predict the evolution of criminal groups, and so on.

In this work, we propose the Adversarial Social Network Analysis Game (`ASNA game`) framework to study the deceptive behaviors of adversarial nodes in social networks. We formulate the framework as a network game, where a data collector is attempting to reveal true information about nodes in the network while individuals in an adversarial group are attempting to disrupt the process. The `ASNA game` framework can be used to experimentally examine a wide variety of questions, including: 1) To what extent, and how, does the network structure influence the deceptive behavior of adversarial nodes?, 2) What level of incentive should a data collector provide to an adversarial node in order to obtain more useful, accurate data? 3) Does loyalty of individuals to the adversarial group or respect for the hierarchy of the organization play a role in deceptive behavior?

We use the `ASNA game` framework to conduct behavioral experiments using participants recruited from Amazon Mechanical Turk (mTurk). These participants play as members of an adversarial group. Our ultimate goal is to use `ASNA game` framework to propose a model that can predict a node's reporting behavior for a specific network setting.

In this work, we introduce the `ASNA game` framework and then discuss its implementation on mTurk. We then confirm the validity of `ASNA game` framework to study adversarial behavior of people by showing 1) Participants understand their role and aspects of the framework, 2) Participants understand their objective in a game and, 3) Participants act as members of the adversarial group.

4.1 Background

Many studies on adversarial behavior come from the criminology, psychology, and sociology literature, and are conducted on criminals under interrogation. Hartwig et al. [44] conducted an analysis to compare strategies used by guilty suspects vs innocent suspects used in a police interrogation. This analysis revealed that guilty suspects used various strategies to appear truthful while innocent suspects revealed the truth showing their innocence. Separately, Hartwig et al. [45] identified the stage at which interrogation evidence should be displayed to detect deception. This analysis showed that revealing evidence later in the interrogation triggered inconsistencies in the stories provided by the criminals revealing their status. Stromwall et al. [94] studied lie-telling strategies by people with criminal experience. Participants showed a great variation of preferred lying but belonging to three main categories: close to the truth, not giving away information, and no strategy. Although such studies are plentiful, they generally do not consider network structure. However, as shown by Galeotti [36] and Wong et al. [110], criminals usually form networks to be able to operate and adjust their networks to hide information.

Recent research on covert network analysis has looked specifically at applying social network analysis to analyze dark networks [67, 20, 56]. Others have considered using social network analysis techniques to disrupt criminal networks [89, 92]. These works mainly consider standalone criminal networks which are not embedded into regular social network, and work under the assumption of correct information. Wijegunawardna et al. [108] propose algorithms to identify criminals in a dark network under the deceptive behavior of individuals. Even though they consider possible inaccuracies in data, they use synthetic models to simulate these inaccuracies. The goal of the present work is to provide such algorithms realistic models of adversarial behavior.

We conduct our experiments using participants hired from Amazon Mechanical Turk (AMT). Recently, AMT has become popular among researchers running human experiments due to the low cost of and easy access to large subject pool [71]. Paolacci et al. [80] show that results obtained from running experiments on AMT are comparable to that of running lab experiments. However, AMT experiments can suffer from bots acting to be human subjects. Mason et al. [71] suggest using filtering checks to ensure that mTurkers understand and pay attention to experiment details. There have not been many works on conducting network experiments on AMT. Suri et al. [97] conducted a networked public goods game experiment on AMT to evaluate the effect of network structure in human cooperation. Our work focuses on studying adversarial behavior of nodes in a network using AMT participants.

4.2 Adversarial Social Network Analysis Game

We model the `ASNA game` as a variation of the popular Werewolf/ Mafia party games [16], in which a minority of players collude against the majority. In the `ASNA game`, a pack of werewolves (`Red` nodes), representing members of an adversarial group, is hiding among innocent civilian villagers (`Blue` nodes) in a village social network, as shown in Figure 4.1. (For ease of reading, in this chapter, we use

`Black` nodes to represent `Red` nodes and `White` nodes to represent `Blue` nodes in the figures. In our actual experiments, their true colors are red and blue.) A sheriff (data collector) begins an investigation to identify the werewolves, who attempt to evade detection. In the current implementation, participants play as werewolves only.

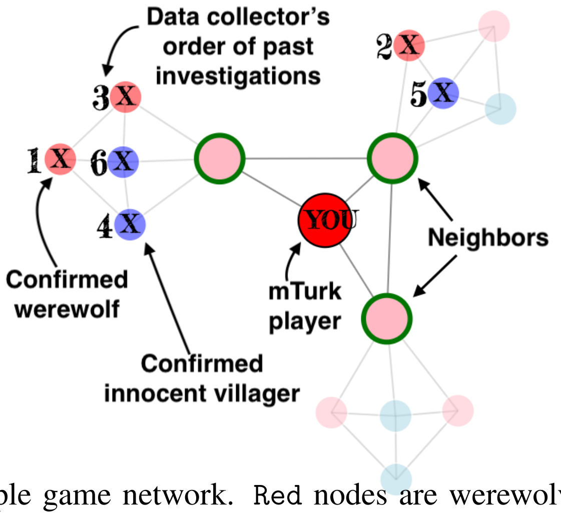
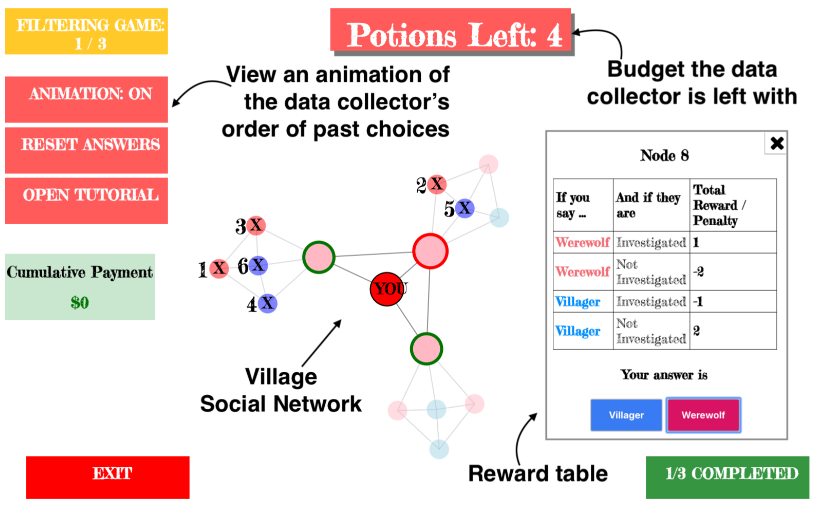

Data collector's order of past investigations
3X
1X
6X
4X
2X
5X
YOU
Neighbors
Confirmed werewolf
mTurk player
Confirmed innocent villager

Figure 4.1: An example game network. `Red` nodes are werewolves and `Blue` nodes are villagers. Dark colored nodes have already been investigated. The numbers by the investigated nodes show the order of investigator past choices. mTurk participant plays as the "YOU" node in the network. Participants need to report colors of the green circled neighbors of the "YOU" node in the network.

Overview of Gameplay: Currently, the game is a single player game, though future versions will allow for multiple players. Each participant (mTurkers) plays as a "YOU" node, a member of the adversarial group, as shown in Figure 4.1. The "You" node has just been investigated by the data collector, and the participant must thus decide whether to report each of her uninvestigated neighbors as `Red` or `Blue` nodes. Each choice is associated with potential rewards and penalties. She may choose to protect some members of the werewolf pack by claiming that they are villagers, and in doing so receive a reward from the pack; but if the data analyst subsequently investigates that neighbor and realizes her lie, she will be penalized. In contrast, if she betrays another werewolf, the pack will penalize her but the data analyst may reward her. To maximize her overall payoff, she must attempt to predict whether a node will subsequently be investigated by the data analyst.

UI: In this version of the game, the player sees the full network structure, including the village's social network, the order of data collector's past choices, shown as numbers beside nodes that are already investigated, and the budget the investigator is left with, which we refer to as potions, in the game UI, as shown in Figure 4.2. At the beginning of each game, players see an animation showing which other individuals have been investigated before the player. This helps the player to determine the data collector's strategy.

Using this information, the player must decide whether to report each neighbor as a werewolf or a villager. Players report each neighbor as a `Red` or `Blue` node by clicking the Werewolf or Villager button in the rewards table.



Node 8

If you say ..	And if they are	Total Reward / Penalty
Werewolf	Investigated	1
Werewolf	Not Investigated	-2
Villager	Investigated	-1
Villager	Not Investigated	2

Your answer is

Villager Werewolf

Figure 4.2: User interface the participants would interact with to report their answers about neighbors. The rewards table opens when a participant clicks on a neighbor and shows possible rewards she can earn for each possible answer. Potions left indicates how many other nodes, the data collector can investigate after the participant.

Rewards and Penalties: To see the rewards and penalties associated with the possible responses about a neighbor, the player clicks on that node. This action shows a matrix similar to that shown in Table 4.1. Rewards and penalties are dependent on 1) the true colors of the neighbors, 2) Whether a participant decide to report the truth or lie about neighbor's color and, 3) whether the data collector ultimately investigates the neighbor. This last factor is not known to the player- he or she must attempt to predict it.

Participants earn a reward from the werewolf pack for reporting false information about a `Red` node- i.e., if they lie to protect a fellow werewolf. Similarly, they get a penalty if they report the truth- i.e., betray a fellow werewolf. On the other hand, participants earn a reward from the data collector if they report the true colors of a werewolf, but get a penalty for lying. Importantly, the data collector awards rewards and penalties only if he subsequently investigates neighbors and confirm whether the participant has lied or told the truth. Thus, the player must attempt to predict the data collector's future actions when deciding her responses. The total reward a participant earns in a game converts to the bonus payment she gets for participating in the experiment. The specific values of rewards and penalties vary depending on the conditions being tested, and correspond to total payouts of roughly \$0.10 - \$0.70.

		Participant			
		Red Neighbor		**Blue Neighbor**	
		Truth	**Lie**	**Truth**	**Lie**
Investigator	**Investigates**	+ reward	- penalty	+ reward	- penalty
	Doesn't investigate	0			
Adversarial Group		- penalty	+ reward	0	

Table 4.1: Possible rewards and penalties participants can earn from the data collector and the adversarial group.

Data Collector: In our current experiments, we assume that the goal of the data collector is to investigate, and thus identify, as many Red nodes as possible. The data collector follows some strategy when selecting individuals to investigate (e.g., selecting those who are connected to to the most known adversarial nodes, randomly selecting nodes, etc.). The data collector has some limit b on the number of nodes that she can investigate. Following the Werewolf game, we refer to each investigation as a "potion". As described above, to incentivize correct responses, the data collector provides penalties or rewards in response to false or true query responses. In our experiments, the data collector is controlled by a bot, and varying the strategy and incentives is part of our experimental conditions that we control.

Goals: The player's task is to decide what to report about each neighbor's color. Presumably, players will generally be motivated by the goal of maximizing their payoff, which corresponds to actual money. However, in some cases, players may show behaviors that are not optimal for this goal. For example, participants may decide to lie about civilian neighbors in order to mislead the investigation and protect more red neighbors, even if this does not lead to the greatest payoff.

In the tutorial preceding game play, to help participants make decisions, they see the following guidelines: 1) They should consider the number of potions left, because this plays a role in the likelihood that the data collector will investigate a particular neighbor, 2) They should attempt to identify any pattern in the data collector's past strategy, because this can help predict which nodes the data collector will investigate next, and 3) They should evaluate the rewards and penalties associated with each choice. The players need to pay attention to how likely it is that she will get caught lying to the investigator when deciding what to report.

Current Rules: The framework allows for many different rule sets. In the current version, the data collector iteratively selects individuals from the network (in accordance with some query strategy) for investigation, and (1) In each query, the data collector determines whether the node is `Red` or `Blue` with perfect accuracy, (2) When the data collector queries the selected node as to the colors of its neighbors (i.e., asks the individual under investigation whether her friends are werewolves or civilians), the queried node may lie.

MTurk Implementation: We implement the `ASNA game` on Amazon Mechanical Turk. In our current experiments, only one member of the adversarial group is played by a human mTurker; the other players are controlled by bots. These bots are placeholders for rest of the nodes in the village social network and do not add any information to the game play in the current version of the game. mTurkers first participate in a tutorial which explains the game interface, gameplay, the stakeholders in the game, and the player's role, as well as giving example games. After completing the tutorial, players take part in three filtering games, and must pass at least two in order to continue to validate that they understand their role.

4.3 Experiments and Results

In this section, we demonstrate the validity of the `ASNA game` by showing that players 1) Pay attention to key elements of the `ASNA game`, 2) Understand the objectives of the game and, 3) Show loyalty to the `Red` group. We set up the game so that at least 20 people participated as the "YOU" (`Red`) node in each experiment. Participants are able to take part in any number of experiments after completing the tutorial and passing the filtering exams.

4.3.1 Participants pay attention to key elements of the ASNA game

There are three key elements to the `ASNA game` that participants need to pay attention to in order for results to be valid: 1) The data collector's past choices 2) How many other individuals the data collector can investigate after the current participant and, 3) The rewards and penalties that the participant would get for each answer choice. We use three filtering games to ensure that participants understand and pay attention to these factors.

In the first filtering game, we test if participants pay attention to whether there is a pattern to data collector choices. We use a path network with the data collector investigating one node at the time, in the order listed on the path. Given this setting, the data collector's next choice is likely to be the participant's neighbor. If

	Passed	Passed only one	Passed only two	Passed all three
Filtering 1: Data collector choices The player must accurately predict that the data collector will investigate his/her neighbor next based on data collector's pattern of choices	63%	25%	36%	29%
Filtering 2: Budget left Enough budget to investigate all nodes in the network	55%			
Filtering 3: Reward table Rewards and penalties are such that the player will get a higher reward by reporting the true colors.	69%			

Table 4.2: Percentage of participants who passed each filtering game.

the participants are paying attention to the details of the game, they should always report the true color of the neighbor.

In the second filtering game, the data collector has enough query budget to investigate every node in the network. Therefore, participants should always report correct colors of the neighbors.

In the third filtering game, we check whether participants consider rewards and penalties when providing their answers. We allocate rewards and penalties to neighbors such that they would always get a higher reward by reporting true colors regardless of whether the data collector investigates them or not.

Table 4.2 shows statistics about how many participants passed each filtering game. If participants were to guess answers fully at random, there would be a 50% chance, 25% chance and 25% of passing filtering game 1, 2, and 3, respectively, and the probability of passing at least two filtering exams is 0.25. However, 211 mTurkers completed all three filtering games, and 139 (66%) passed at least two filtering games, and moved on to the next stage of the experiment. These filtering games help ensure that participants understand the game.

4.3.2 Participants understand objectives of the game

To further assess whether participants understand the game, we performed three experiments to evaluate whether the participants try to maximize their reward in the game. We designed experiments corresponding to this hypotheses, as follows:

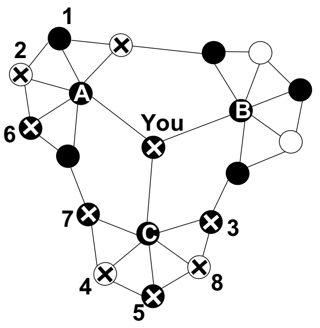
Hypothesis 1: *People are more likely to report false information if they think that the data collector will not discover the truth.*

The `ASNA game` framework can convey information about the risk of the data collector uncovering the truth through the animation showing the data collector's past investigations. For example, if the data collector past choices seem to be random, she poses a little threat to the participant as compared to the case where the data collector is making choices near participant's neighbors. We use three different settings to test this hypothesis.

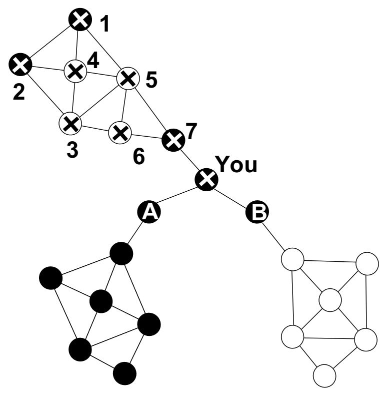
Setting 1: The data collector follows a clear pattern of investigation vs. a random selection of nodes

If the number of potions remaining is small compared to the number of un-investigated nodes, we expect to see people misreport colors of their neighbors more often when the data collector has made random choices in the past as compared to when the data collector has followed a clear pattern of investigation that is likely to lead to the participant's neighbors. We use two networks to test this hypothesis. In one network, the data collector investigates nodes selected at random, and in the other network, the data collector investigates nodes row by row in the network. Rewards and penalties are the same for respective nodes in both networks. We randomly assign participants to one of the two networks to evaluate whether there is a difference in likelihood of misreporting colors between the two networks.

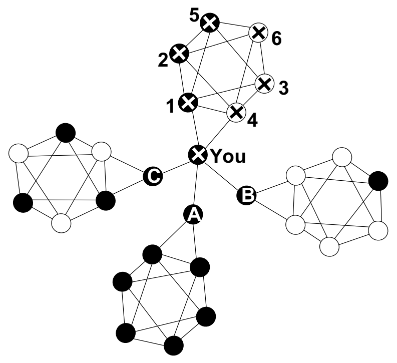
Participants reported false information about `Red` nodes twice as often in the random choice network as compared to when the data collector is following the row-by-row pattern. We confirm using a one tailed t-test at $p = 0.05$ significance that the participants on average reported false information about `Red` nodes in the random choice network significantly more compared to when the data collector follows some pattern.



(a) Percentage of neighbors investigated



(b) Bridge between Red and Blue groups



(c) Protecting neighbors leading to more red nodes

	# participants	A	B	C	A, B	B, C	A, C	A, B, C
% of nbrs investigated	24	$54\%^{*,\dagger}$	$67\%^{*,\ddagger}$	$38\%^{\dagger,\ddagger}$	46%	29%	33%	29%
Bridge between `Red` and `Blue` groups	20	$50\%^{*}$	$15\%^{*}$		5%			
Protect nbrs leading to more `Red` nodes	36	$39\%^{\dagger}$	$42\%^{\ddagger}$	$31\%^{\dagger,\ddagger}$	31%	25%	25%	31%

Figure 4.3 & Table 4.3: Network a) tests whether the likelihood of adversarial nodes misreporting data changes (based on how participants perceive the likelihood of the data collector discovering the truth) and, networks b) and c) test whether participants show loyalty to the `Red` group. Table shows the percentage of participants, who have misreported information about their neighbors in the networks shown. Statistically significant differences from a one tailed t-test at 0.05 significance are marked with $*, \dagger, \ddagger$ symbols. Cells with the same symbol represent that the larger value is significantly larger than the smaller value.

Setting 2: Data collector investigates some parts of the network more compared to others

In this setting, we test Hypothesis 1 by using the network shown in Figure 4.3a. The data collector has investigated each neighbor's neighborhood to a varying degree. We expect participants to identify those neighbors whose neighborhoods have been investigated less thoroughly, and thus provide false information about those neighbors more often. We keep rewards and penalties the same for all three neighbors.

Participants reported that C is not a `Red` node more often compared to A and B, even though all of them are indeed `Red` nodes. The only difference between these three neighbors is in the fraction of their neighborhoods that have been investigated. Using a t-test we confirm that likelihood of misreporting B is significantly lager than the likelihood of misreporting A and C at the $p = 0.05$ significance level.

Setting 3: All neighbors have similar anticipated rewards
We created another network to evaluate how participants balance between risk and reward. In this network, participant has there neighbors A, B, and C who are at high, medium, and low risk of being investigated, respectively. Moreover, they have rewards and penalties of $\pm 5, \pm 3$, and ± 1, respectively. In other words, the higher the risk, the better the potential reward and the worse the potential penalty. The ultimate payoffs would be similar for all three neighbors when we consider the risk vs reward. Therefore, if participants are trying to maximize the reward they would earn, they should lie equally about all three neighbors. Experiments conducted using this network confirm this hypothesis since participants have not lied about any of the three neighbors significantly more than any other neighbor.

These three experiments confirm our hypothesis that participants understand their objectives in the game. Therefore, the experiments validate that the `ASNA game` framework can be used to understand deceptive behavior of adversarial nodes.

4.3.3 Participants are loyal to the Red group

Even though we want to study behavior of adversarial nodes, our experiment subjects consists of workers from AMT, who may not be people with an inherent adversarial mindset. However, even though the participants may not themselves be criminals, the framework allows us to study aspects of general human behavior.

One trait of interest is that of members of some group is being loyal to that group.

Hypothesis 2: *If nodes show loyalty to their group, they would take risks and report false information about neighbors to protect members of their own group*

We test this hypothesis in two different settings. In the first setting, we position the player as a bridge between a `Red` group and a `Blue` group, to see if she tries to direct the data collector towards the `Blue` group and protect the `Red` group. In the second setting, we test whether the size of the `Red` group matters to the participant in the decision process, with players preferring to protect larger `Red` groups.

Setting 1: Bridge node between neighbors leading to a `Red` group and a `Blue` group

Figure 4.3b shows the experiment we designed to test Hypothesis 2. A and B both have same rewards and penalties, and the investigator order of past choices doesn't provide any useful information about which of them would be investigated next. Participants reported false information about A three times more often. According to the network structure, reporting the truth about A would lead the investigation towards a large `Red` compared to B. A t-test with $p = 0.05$ significance level confirms that participants are likely to provide false information about A significantly more often compared to B. This shows that participants are loyal to the `Red` group even if this does not affect their reward.

Setting 2: Protect larger `Red` group

We formulate the network in Figure 4.3c. The "YOU" node is adjacent to nodes A, B, and C, which lead to six, three, and one `Red` nodes, respectively. Rewards and penalties are the same for all neighbors. Table 4.3 shows that participants lied about nodes A and B significantly more compared to node C (significant at $p = 0.05$). However, we do not see a significant difference between the likelihood of lying about A, compared to B. This may be due to participants perceiving that both A and B would lead to similar `Red` groups. Since participants have lied significantly more about A and B compared to C, we can still conclude that participants try to protect neighbors that lead to a larger `Red` group. However, we cannot quantify the how large the `Red` group should be to observe such behavior.

These two experiments show that participants try to protect the `Red` groups even if that doesn't necessarily maximize their objective. We can conclude that participants are loyal to the `Red` group and try to protect the members since they act as members of the group.

4.4 Discussion and Conclusion

We propose the `ASNA game` framework to study adversarial behavior of nodes in a social network. The `ASNA game` framework is modeled as a network game played between a data collector and members of an adversarial group. By varying aspects of the game, we evaluate how the network structure, rewards and penalties, and data collection behavior influence adversarial behavior. Initial analysis using Amazon Mechanical Turk shows that 1) Participants understand their role in the game, and 2) Participants show loyalty to the group. Findings from this type of analysis may be helpful in designing network analysis algorithms that are robust to targeted misinformation, or in understanding the behavior of covert groups in general.

One drawback of our current work is that we recruit general workers from AMT to participate, rather than criminals. Thus, their mindset is different, creating different behaviors. However, we show that even with these workers, participants are loyal to the `Red` group and provide misinformation to the data collector to protect `Red` nodes showing that they in fact act adversarial towards the data collector without any necessary gain in rewards.

We intend to extend the present work by exploring other factors, such as the stated hierarchical ranks of other nodes, the existence of herding behavior, tit-for-tat behavior, more variations on network structure, how rewards and penalties can affect node behavior and so on. Additionally, we will extend the current work to a multi-player setting, allowing us to better understand group dynamics.

Even though the `ASNA game` framework is designed to understand behavior of adversarial groups towards a data collector, findings from such a behavior analysis can be used target minority groups who are trying to hide from an authoritative figure such as a government. Therefore, when using this framework to study adversarial behavior, careful attention should be made to only test hypothesis that are relevant to criminal groups or covert entities. For example, best use of this framework would be to study what criminal network hierarchies are more adversarial, how various network structures and criminal network hierarchies can affect adversarial behavior.

Chapter 5 Predicting People of Interest with Bounded Error Rates

Locating POIs in social networks is closely related to the node classification problem in social networks. Node classification algorithms predict labels for unlabeled nodes in a partially labeled graph by using known node labels and connections between nodes. For example, consider a criminal group hidden inside a general social network. If some criminals and non-criminals are identified, can an algorithm predict whether the unlabeled nodes are criminals? If a company wants to target an advertisement to students at a particular university in an online social network, and the universities of some nodes are known, can the university of the other nodes be inferred? Properties of such POIs are not necessarily correlated with the network structure. In such cases, node classification algorithms play an important role: By taking advantage of connections as well as node attributes, algorithms specifically designed for node classification generally perform better on semi-supervised graph classification tasks as compared to traditional machine learning algorithms [117, 76].

In many problems, the user of a node classification algorithm may wish to as-

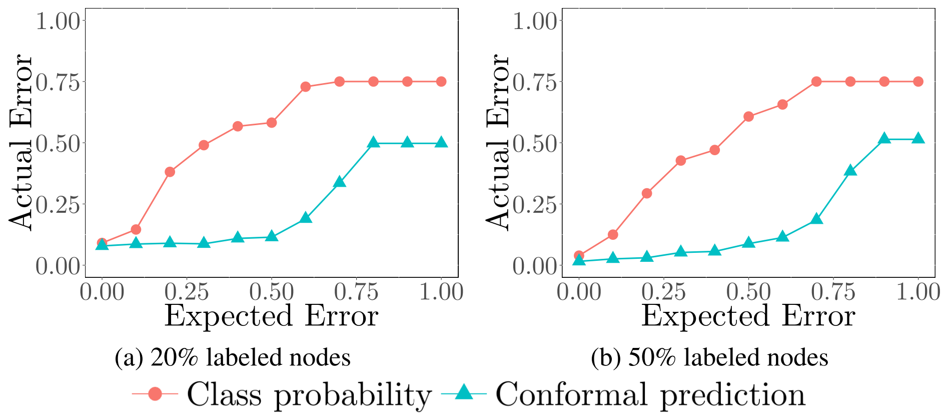


Figure 5.1: Comparison of using conformal prediction framework vs class probability as expected error for Cora citation dataset classification using Iterative Classification Algorithm. Conformal prediction actual error is always no greater than the expected error where as the actual error for class probability does not follow the expected error.

sociate a confidence with each prediction. For example, consider the dark network data collection problem in Chapter 3, where we intend to sample a dark network to maximize the number of identified covert entities in the sample. Here, we predict whether a node in a social network is a criminal or not. A prediction in this problem may lead to a real-world criminal investigation. It is thus useful to know how likely is a prediction is to be wrong. In such applications, prediction algorithms that can provide guaranteed error rates on unseen data are essential.

The performance of a node classification algorithm is generally measured with metrics such as accuracy, precision, and recall. Such measures describe the algorithm performance in aggregate, but do not measure certainty of individual predictions. While node classification algorithms can generally output a vector indicating the probability that a node belongs to each class, as Figure 5.1 shows, these values should not be interpreted as confidence values. In this figure, when using these probabilities as expected errors, we see that the actual errors are larger than the expected errors. In contrast, the conformal prediction framework actual errors are always lower than the expected error.

In this work, we demonstrate how the *conformal prediction* framework can be used to obtain error bounds for the node classification task. Conformal prediction (CP) is a framework to provide guaranteed error bounds for prediction algorithms [91]. This framework works on top of a prediction algorithm (e.g. SVM, Neural Network) and for a specified error bound, considers how unusual a data instance is in consideration to training data. The CP framework has been applied to provide guaranteed error bounds for machine learning algorithms [70, 51, 81]. However, to our knowledge, conformal prediction has not been applied to the network setting. Figure 5.1 shows an example application of the CP framework to node classification problem. The prediction error rate is always lower then the expected error the CP framework suggests.

We consider node classification algorithms from different categories and show how the CP framework can be applied to obtain guaranteed error rates for these algorithms. Further, we conduct an experimental analysis over various types of node attributes and graphs and show that the CP framework can improve node classification algorithm accuracy.

5.1 Background

We first provide relevant background on node classification algorithms, and then explain the Conformal Prediction Framework and related work.

5.1.1 Node Classification Algorithms

Node classification algorithms consider both node attributes and node connectivity patterns when making predictions. There are three main categories of node classification algorithms. The first category contains local classifier based algorithms, where a local classifier is iteratively trained using node attributes and network information to predict labels for unlabeled nodes, such as logistic regression local classifier. Iterative Classification Algorithm (ICA) [76] and Link Based Classification algorithm [66] are examples of such algorithms. These algorithms iteratively predict labels for the unlabeled nodes in the graph using predicted labels in the round of predictions.

The second category of algorithms are label propagation based algorithms, where the algorithms use random walks to learn a global labeling function across the network [117]. These algorithms predict labels for nodes in the graph by considering hitting probability of each label in a random walk.

The third, and newest, category of node classification algorithms learn a deep representation of the network and labeling function. There are two approaches to learning this representation. The first approach uses network embedding-based algorithms, which generate feature vectors for nodes in a graph in an unsupervised manner. These algorithms use multiple random walks starting at each node, and trains a prediction model based on these features [98, 83, 42]. The second approach is learning a labeling function using deep neural networks based on graph representation. Graph Convolutional Neural Networks (Graph CNN) are widely used to conduct node classification under this category [54].

In the current work, we consider node classification algorithms from each category mentioned above and show how the `CP` framework can be applied to obtain predictions with guaranteed error bounds.

5.1.2 Conformal Prediction Framework

The `CP` framework outputs a set of predictions for a given sample with a bounded error rate by comparing "how typical" the sample is as contrasted to other samples [91]. Suppose that we are given a data set $Z = \{z_1, z_2, \ldots, z_n\}$ where $z_i = (x_i, y_i)$; $x_i \in \mathbb{R}^d$ is the feature vector of the sample i, and $y_i \in Y$ is the class label for i. Here, Y is the set of class labels, i.e. $Y = \{y^1, y^2, \ldots, y^\ell\}$.

Given a new sample with feature vector x_{n+1}, the CP framework measures how typical the following sequence is: $(z_1, z_2, \ldots, z_n, (x_{n+1}, y^k))$, where $y^k \in Y$. Since we already know the labels for $z_1, z_2, \ldots, z_n$, we are in effect measuring how typical the sequence is when label y^k is assigned to the new sample, and how likely is that $n+1$'s true label is y^k [81].

The CP framework uses a test for randomness to measure how likely a sequence is, where the p value for a given sequence is calculated using Equation (5.1). A given conformity measure calculates the "typicalness" of a data instance (α values). The α_i is the conformity score for i^{th} data instance [81].[1]

$$p(z_1, z_2, \ldots, (x_{n+1}, y^k)) = \frac{|\{i = 1, \ldots, n : \alpha_i \leq \alpha_{n+1}\}|}{n} \tag{5.1}$$

The CP framework calculates the p-values for all sequences considering all possible class labels. Given some significance value ϵ, prediction set of $n+1$ instance at ϵ significance is then calculated using Equation (5.2).

$$P(n+1, \epsilon) = \{y^k : y^k \in Y \quad \& \quad p(z_1, z_2, \ldots, (x_{n+1}, y^k)) > \epsilon\} \tag{5.2}$$

For example, if we set significance to 0.05 and some label has a p-value less than 0.05 meaning that the chance of generating the sequence including the label in consideration is less than 5%. This implies that the label we are considering is highly unlikely.

Note that in Equation (5.2), the CP framework outputs the set of labels that satisfy the specified significance rather than a single prediction. Therefore, CP framework predictions can have one prediction, multiple predictions, or zero predictions, in case none of the labels satisfy the significance requirement. The probability of not including true labels in the prediction set is less than the specified

[1]Note that the "$\leq$" sign in Equation 5.1 changes to "$\geq$" if we are using a non-conformity function instead of conformity.

threshold, providing an error rate bounded by the significance level. If we are to predict labels at significance ϵ, the probability of not including the correct label in the prediction set is ϵ and the confidence in the prediction is $1 - \epsilon$.

The CP framework provides guaranteed error bounds for predictions under the assumption that the data is exchangeable, meaning any permutation of the sequence $(z_1, z_2, \ldots, z_n, (x_{n+1}, y^k))$ should result in the same p-value. This assumption is necessary to obtain the p-value using Equation (5.1).

The CP framework was originally introduced in the *transductive* setting, where the true label of the current sample is revealed before the arrival of the next sample [91]. In this setting, the given model is trained considering each possible label for new data instance and the framework measures how typical the model is. Since this setting requires training the model for each new data instance and each possible label, applying this in a real world setting would be very inefficient.

The *Inductive Conformal Prediction* (ICP) is an alternative approach which splits the training data into actual training set and a calibration set, and uses the calibration set to conduct CP [81]. The ICP framework uses the training set to train the underlying prediction model, and the calibration set to calculate the p-value. In the ICP setting, we only consider the calibration set when calculating the p-value of Equation (5.1).

5.1.3 Related Work

Bayesian Framework and Probably Approximately Correct Learning theory (PAC theory) [46] are two other frameworks that provide bounded error rates in machine learning applications. Bayesian Framework error rates are dependent on the priors that are used in the estimation. Hence, the error bounds are not guaranteed in case priors are wrong. PAC theory provides upper bounds on prediction algorithms rather than individual samples. Furthermore, the error bounds provided by PAC theory are heavily dependent on the quality of the data [81]. The only assumption that the CP framework makes is that the data is exchangeable, which is valid for most machine learning data. Dashevskiy et al. [21] show that even in cases where exchangeability assumption is violated (e.g., time series data), the CP framework still provides reasonable error bounds. The CP framework, unlike PAC theory, can provide error bounds for individual samples rather than the algorithm.

Cross Conformal Prediction (CCP) is another variation of CP framework. The CCP is very similar to ICP, with the difference that rather than selecting a part of the training data as calibration data, CCP follows a cross-validation-like approach to provide error bounds. Similar to cross validation, CCP leaves out one part of training data as calibration data, and other parts are used to train the model. This process continues for all folds of the training data, and outputs the average error bound for a sample. In this work we show how ICP can be applied to obtain error bounds for node classification problems. The CCP can similarly be applied to node classification problem.

Initial work on the CP framework was primarily theoretical, and focused on proving the error bounds. Applying the CP framework to machine learning algorithms required defining conformity measures specific to algorithms, showing that the data is in fact exchangeable. Research in this area shows how the CP framework can be applied to various algorithms including decision trees [51], neural networks [81], SVM [70], time series prediction [21], random forest [26], and regression [82] etc.

To best our knowledge, this is the first work that considers providing guaranteed error bounds for node classification algorithms, and shows how the CP framework can be applied to obtain those error bounds

5.2 Methodology

We now introduce the details of how the `ICP` framework can be applied to node classification algorithms. In the $z_n = (x_n, y_n)$ a node classification problem, we have that $x_n \in \mathbb{R}^d$ is the d-dimensional feature vector for node n, and y_n is the label of node n.

To show that the `ICP` framework applies, we must demonstrate that the data is exchangeable. Note that this does *not* require that the data is i.i.d., simply that all permutations of each sequence are equally likely. Since we are drawing training and calibration samples uniformly at random from the set of nodes, exchangeability holds. We also considered sampling training and calibration data using a network crawling algorithm such as random walk or snowball sampling. Resulting error bounds are not valid in these cases since any training node ordering is not equally likely (not exchangeable) for random walk or snowball sampling.

For example, consider we start a random walk from node A in Figure 5.2 and sample nodes to assign initial label for a node classification task. A possible random walk sample is $S = \{A, B, C, D, E\}$. In order for `ICP` predictions to provide bounded error rates, any ordering of S should be equally likely. But, the ordering $S' = \{D, B, A, C, E\}$ is not possible in a random walk, since a walk cannot move directly from D to B. However, if we sample S uniformly at random, any node ordering is equally likely.

In many real world applications, we begin with an unlabeled graph and obtain labels for some of the nodes, in order to label rest of the nodes using a semi-supervised node classification algorithm. In such cases, we can select a random node sample to label to facilitate using `ICP` to get bounded error rates. In cases where we have to use a network crawling algorithm, Metropolis Hasting random walk [95] and Re-weighted random walk [85] are two example algorithms that provide approximately uniform node samples.

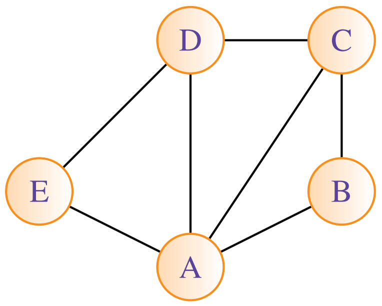


Figure 5.2: Sampling algorithm exchangeability example.

The conformity function is an integral part of the ICP framework, measuring how different the data instance in consideration from the calibration set. Any real valued conformity function that measures how different a sample is can be used to produce valid nested prediction regions [103], but the efficiency (smaller prediction regions) of the algorithm depends on how well the nonconformity function measures differences between data instances. Consider a prediction algorithm that outputs a vector $\sigma_n \in \mathbb{R}^{|Y|}$ for some unlabeled node n, indicating the probability that n would belong to each class in Y. One possible conformity measure for such an algorithm is the probability margin, which is the difference between the label in consideration and the highest probability of any other label [88]. We can calculate the probability margin conformity score for some label $y^k \in Y$ using Equation 5.3.

$$C(n, y^k) = \sigma_n(y^k) - \max_{y^i \in Y : y^i \neq y^k} (\sigma_n(y^i)). \tag{5.3}$$

Given a node classification algorithm M, a graph G, set L of labeled nodes, set U of unlabeled nodes, a significance level ϵ, and a conformity function C, we introduce Algorithm 4 to show how the ICP framework can be applied to node classification problem.

Algorithm 4 ICP for Node Classification

```
Input: $G$= Graph, $L = labeled\_nodes$, $U = unlabeled\_nodes$,$M$ = prediction
       algorithm, $C$= Conformity function, $\epsilon$ = significance
Output: Prediction set for each node in $U$ at significance $\epsilon$
procedure ICP
    Divide $L$ into $T = training\_set$ and $S = calibration\_set$
    Train $M$ using $G$ and $T$                          ▷ Train prediction model $M$
    for $s \in S$ do
        $\sigma_s = M(s)$                                ▷ Get prediction probability vector $\sigma_s$ for $s$
        $\alpha_s = C(\sigma_s, y_s)$                    ▷ Calculate conformity score for $s$ and $s$'s label $y_s$
    for $u \in U$ do
        $P_u = \{\}$                                     ▷ $u$'s prediction set at significance $\epsilon$
        for $y^k \in Y$ do
            $\sigma_u = M(u)$
            $\alpha_u = C(\sigma_u, y^k)$
            $p = \frac{|\{s \in S : \alpha_s \leq \alpha_u\}|}{|S|}$          ▷ Calculate p-value for label $y^k$
            if $p > \epsilon$ then
                $P_u.add(y^k)$                           ▷ Add $y^k$ to $u$'s prediction set
```

5.3 Experiments

We conduct experiments to evaluate whether the ICP framework predictions meet the specified error bounds. We consider node classification algorithms from different categories: Iterative Classification Algorithm (ICA), Label Propagation (LP), Graph Convolutional Network (GCN), and DeepWalk (DW). We now introduce the perfromance metrics and data sets used in our research.

5.3.1 Performance Metrics

Our evaluation closely follows the evaluation criteria in [51]. We use several measures to evaluate the quality of predictions made by ICP framework for the node classification problem:

1. We check whether the ICP framework predictions meet the specified maximum error bounds. Since node classification graph data meets the exchange-

ability assumption, the specified error bounds should be met.

2. We evaluate the `ICP` framework predictions based on their efficiency. Since the ICP framework outputs a set of predictions for a node based on its conformity score, an efficient prediction would have only a single class in the prediction set. We consider the fraction of predictions with only one class (OneC), multiple classes (MultiC) and zero classes (ZeroC) to evaluate the efficiency of the ICP framework.

3. We compare the accuracy of the baseline prediction model (BaselineAcc) with the accuracy of one class predictions (OneAcc) from the ICP framework to show that the ICP framework enhances performance of the baseline prediction model.

5.3.2 Datasets

Node classification algorithms generally perform well on assortative networks, but less well on nodes are not assortative. Accordingly, we have selected graph datasets with varying levels of assortativity. For each network, we use the largest connected component. Cora [74, 90] and PubMed [75] are citation networks, showing citation relationships between papers. Facebook100 [2] is the Amhrest college Facebook friendship network. BlogCatalog [114] is a blogger friendship network. The Protein-Protein Interaction network [42] is a subgraph of the PPI Homo Sapiens network. Networks are described in Table 5.1.

5.3.3 Experimental Setup

We run experiments as a multi-class prediction problem where we vary the percentage of labeled nodes in the network from 10% up to 50%. We randomly sam-

[2]Obtained from https://archive.org/download/oxford-2005-facebook-matrix.

Dataset	**Type**	**Nodes**	**Edges**	**Label**	**Classes**	**Label Assortavity**
Cora	Citation	2708	5278	Research area	7	0.771
PubMed	Citation	19717	44327	Research area	3	0.686
Blogcatalog	Social	10312	333983	Blogger group	39	0.05
Facebook100	Social	2235	90954	Year	9	0.409
PPI	Biological	3890	38739	Biological state	50	0.05

Table 5.1: Network dataset statistics

ple the labeled data from each class proportional to the size of the class and report average performance over 10 runs. We used 25% of the training data as the calibration set to conduct conformal prediction.

For ICA and Deepwalk, we use a multi-class logistic regression classifier as the base classifier. We set Deepwalk hyper parameters for all data sets as follows: 80 walks, 128 dimension representation, window size 10, and walk length 40 according to [83]. GCN hyper parameters are set at 0.5 dropout rate, 5.10^{-4} $L2$ regularization and 16 hidden units, according to [54].

5.4 Results

Figure 5.3 shows results of `ICP` using ICA on the Cora citation network with $10\%, 30\%$ and 50% of the nodes labeled, using the performance metrics discussed in Section 5.3.1. First, we see that the actual errors in all algorithms are very close to the given significance level, demonstrating that the `ICP` framework in fact provides accurate error bounds for node classification algorithms.

Second, as expected, the percentages of OneC (one-class predictions) and MultiC (multiple-class predictions) increase and decrease as we increase the significance level, respectively. At lower significance, there can be many classes that satisfy the given error bounds according to Equation 5.1. ZeroC (zero class predictions) slightly increases at higher significance values causing OneC to reduce

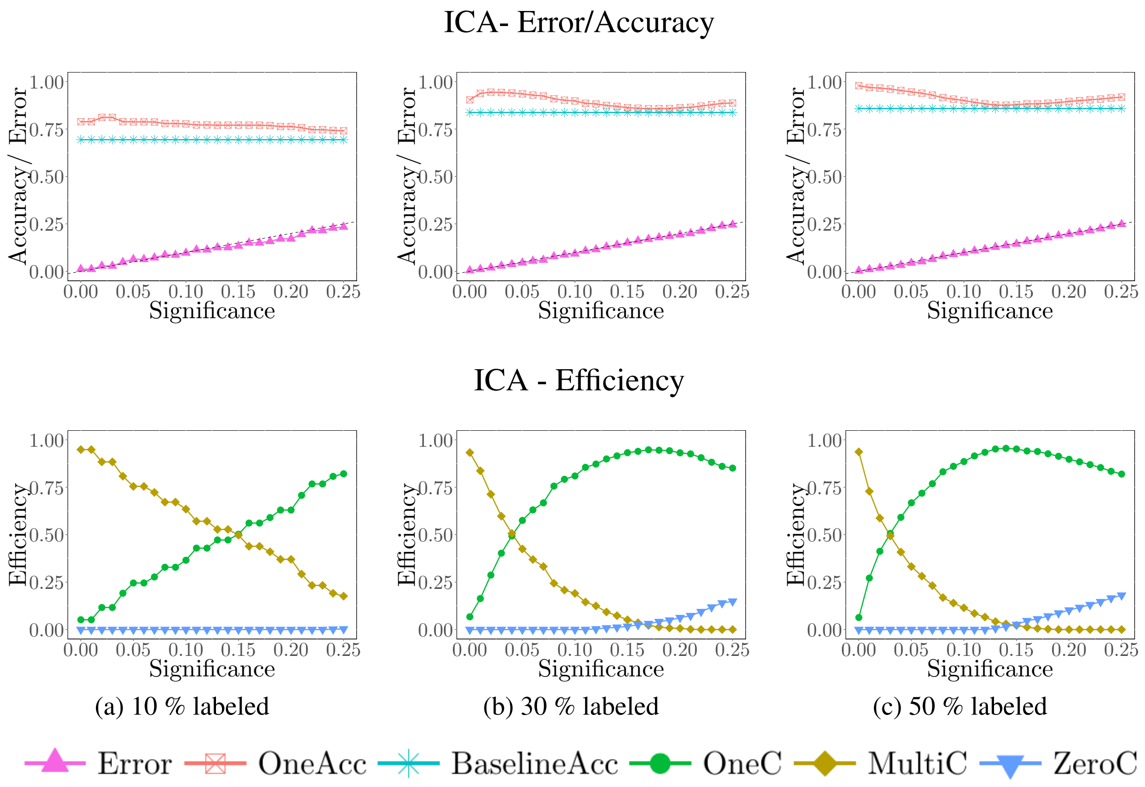


Figure 5.3: Accuracy and efficiency for Cora citation data set using ICA as the baseline algorithm when 10%, 30% and 50% of the nodes are labeled. The actual error is always no greater than the specified significance level.

slightly, because some nodes do not meet significance for any single class label.

Finally, we see that the accuracy of the `ICP` framework is higher than the accuracy of the baseline node classification algorithm, showing that the predictions from `ICP` are more reliable than those from the baseline prediction algorithm. Results are consistent across different algorithms and labeled node percentages.

Tables 5.2 and 5.3 summarize the performance of the `ICP` framework applied to ICA and Label Propagation, respectively. Both algorithms closely maintain the given error bounds. `ICP` framework can cause the prediction errors to be slightly higher than the given error bound since the predictions are based on the calibration set rather than the whole training set.

Further, applying `ICP` improves baseline accuracy of both algorithms in all data sets. The `ICP` improves ICA accuracy in FB100 data from 0.82 to 0.94, while

Dataset	Significance	Error	OneC	MultiC	ZeroC	OneAcc	BaseAcc
Fb100 Amherst	0.05	**0.045 ± 0.01**	0.58	0.42	0	0.94	
	0.15	**0.133 ± 0.02**	0.88	0.12	0	0.87	0.82
	0.25	**0.25 ± 0.02**	0.85	0	0.15	0.88	
BlogCatalog	0.05	**0.05 ± 0.01**	0.05	0.95	0	0.52	
	0.15	**0.15 ± 0.01**	0.11	0.89	0	0.48	0.23
	0.25	**0.25 ± 0.01**	0.15	0.85	0	0.44	
PubMed	0.05	**0.046 ± 0.01**	0.52	0.48	0	0.92	
	0.15	0.154 ± 0.01	0.97	0.03	0	0.85	0.83
	0.25	**0.257 ± 0.01**	0.84	0	0.16	0.89	
PPI	0.05	**0.051 ± 0.01**	0.03	0.97	0	0.21	
	0.15	0.147 ± 0.01	0.08	0.92	0	0.18	0.10
	0.25	**0.248 ± 0.02**	0.14	0.86	0	0.17	

Table 5.2: Conformal prediction framework performance using ICA as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run.

Dataset	Significance	Error	OneC	MultiC	ZeroC	OneAcc	BaseAcc
Cora	0.05	**0.043 ± 0.01**	0.60	0.40	0	0.94	
	0.15	**0.141 ± 0.03**	0.89	0.09	0.01	0.87	0.83
	0.25	0.252 ± 0.04	0.85	0	0.15	0.89	
PubMed	0.05	0.052 ± 0.01	0.54	0.46	0	0.91	
	0.15	**0.149 ± 0.01**	0.92	0.08	0	0.85	0.82
	0.25	0.253 ± 0.01	0.87	0	0.13	0.86	
Fb100 Amherst	0.05	0.052 ± 0.01	0.38	0.62	0	0.93	
	0.15	**0.144 ± 0.01**	0.71	0.29	0	0.88	0.78
	0.25	0.252 ± 0.03	0.92	0.01	0.07	0.81	
BlogCatalog	0.05	**0.049 ± 0.01**	0.04	0.96	0	0.36	
	0.15	**0.149 ± 0.01**	0.10	0.90	0	0.38	0.22
	0.25	0.253 ± 0.01	0.15	0.85	0	0.39	
PPI	0.05	**0.048 ± 0.01**	0.04	0.96	0	0.12	
	0.15	**0.146 ± 0.01**	0.10	0.90	0	0.11	0.10
	0.25	**0.239 ± 0.03**	0.15	0.85	0	0.11	

Table 5.3: Conformal prediction framework performance using Label Propagation as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run.

predicting singleton labels for 58% of the nodes with a guaranteed error rate of 5%. In the Label Propagation algorithm, `ICP` improves accuracy for Facebook100 data from 0.78 to 0.93, while predicting singleton labels for 38% of the nodes with a guaranteed error rate of 5%. When the baseline predictor accuracy is reasonable, `ICP` provides efficient predictions (more singleton predictions). When the base-

Dataset	Significance	Error	OneC	MultiC	ZeroC	OneAcc	BaseAcc
Cora	0.05	**0.045 ± 0.01**	0.70	0.30	0	0.95	0.83
	0.15	**0.141 ± 0.02**	0.93	0.07	0	0.86	
	0.25	**0.248 ± 0.03**	0.83	0	0.17	0.91	
PubMed	0.05	**0.047 ± 0.01**	0.72	0.28	0	0.94	0.85
	0.15	0.151 ± 0.01	0.98	0.01	0.01	0.86	
	0.25	**0.249 ± 0.01**	0.82	0	0.18	0.92	
Fb100 Amherst	0.05	**0.048 ± 0.01**	0.43	0.57	0	0.96	0.72
	0.15	**0.139 ± 0.02**	0.59	0.41	0	0.9	
	0.25	0.251 ± 0.02	0.90	0.10	0	0.77	
BlogCatalog	0.05	**0.049 ± 0.01**	0.001	0.999	0.05	0	0.12
	0.15	**0.139 ± 0.01**	0.005	0.995	0	0.05	
	0.25	0.238 ± 0.01	0.01	0.99	0	0.11	
PPI	0.05	**0.049 ± 0.01**	0.0002	0.9998	0	0.03	0.05
	0.15	**0.143 ± 0.02**	0.002	0.998	0	0.18	
	0.25	0.239 ± 0.02	0.005	0.995	0	0.11	

Table 5.4: Conformal prediction framework performance using GCN as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run.

Dataset	Significance	Error	OneC	MultiC	ZeroC	OneAcc	BaseAcc
Cora	0.05	**0.048 ± 0.01**	0.59	0.41	0	0.94	0.82
	0.15	**0.150 ± 0.02**	0.90	0.09	0.01	0.86	
	0.25	**0.239 ± 0.03**	0.88	0	0.12	0.87	
PubMed	0.05	**0.048 ± 0.01**	0.53	0.47	0	0.92	0.80
	0.15	**0.149 ± 0.01**	0.89	0.11	0	0.84	
	0.25	**0.245 ± 0.01**	0.90	0	0.10	0.84	
Fb100 Amherst	0.05	**0.047 ± 0.02**	0.001	0.999	0	0.14	0.15
	0.15	0.155 ± 0.02	0.005	0.995	0	0.12	
	0.25	0.268 ± 0.04	0.01	0.99	0	0.13	
BlogCatalog	0.05	0.057 ± 0.01	0.012	0.988	0	0.82	0.28
	0.15	0.153 ± 0.01	0.02	0.98	0	0.79	
	0.25	0.255 ± 0.01	0.03	0.97	0	0.76	
PPI	0.05	0.051 ± 0.01	0.0002	0.9998	0	0.43	0.11
	0.15	0.151 ± 0.02	0.005	0.995	0	0.32	
	0.25	**0.250 ± 0.02**	0.007	0.993	0	0.31	

Table 5.5: Conformal prediction framework performance using DeepWalk as the baseline algorithm. Average performance over 10 runs where randomly selected 30% of the nodes are labeled in each run.

line predictor does not perform well, conformity scores also become less meaningful leading `ICP` to make more multiple predictions. In blogcatalog, at significance level 0.15, only 11% of the predictions are singletons.

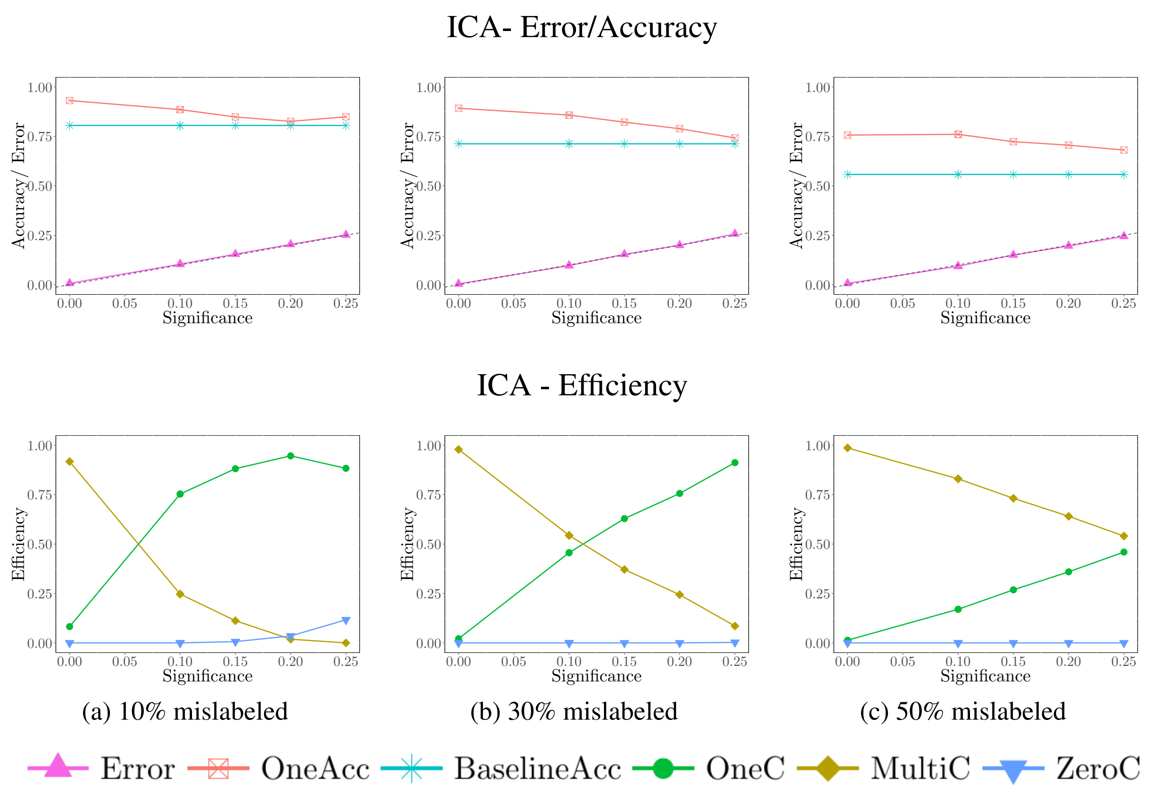


Figure 5.4: Performance of `ICP` applied to CORA citation data with 30% of nodes initially labeled. ICA is used as the baseline. Note that $10\%, 30\%$ and 50% of training data mislabeled. `ICP` maintains the error bounds even when 50% of the training data is mislabeled. But the efficiency decrease as there are more errors.

Tables 5.4 and 5.5 summarize results for GCN and DeepWalk respectively. GCN algorithm works well when node labels show homophily (Cora, FB100 and PubMed). In the Blogcatalog and PPI data sets, GCN algorithm baseline accuracy is 0.12 and 0.05, making it impractical to get meaningful predictions. The Deepwalk algorithm only considers network structure when predicting labels. If node labels are not correlated with the structure, even if data shows high homophily, Deepwalk baseline accuracy is low. In general, both these algorithms maintain the error bounds but provide inefficient predictions in some cases.

5.4.1 Perturbation Analysis

Real world network data collection can be prone to errors. In Figure 5.4, we show the effect of mislabeled data on `ICP` framework predictions. We consider the CORA data set with 30% of the nodes initially labeled and change labels randomly for $10\%, 30\%$ and 50% of the nodes in the training data. Figure 5.4 shows that mislabeled training data does not affect `ICP` error bounds. As we increase the percentage of mislabeled data, the efficiency of predictions decreases, since the percentage of singleton predictions decreases.

5.4.2 Running Time Analysis

The running time of the `ICP` framework is governed by the running time of the baseline prediction algorithm. Given a trained model to predict labels, the running time of the `ICP` framework for a given significance level is $\mathcal{O}(|U| \cdot |Y|)$, where U here is the set of unlabeled nodes, and Y is the set of labels. For each node in the unlabeled node set, we calculate the p-value for each label in Y.

5.5 Discussion and Conclusion

In this work we consider the problem of providing guaranteed error bounds for predictions in node classification algorithms. We use the `CP` framework, which works with a given prediction model to provide bounded error rates. We use `ICP` a more efficient variant of the `CP` framework and show how this can be applied to ICA, Label Propagation, GCN and DeepWalk algorithms to improve prediction accuracy and provide more reliable predictions. We evaluate performance of this framework using citation, social and biological networks and show that $1)$ Specified significance levels are maintained across all data sets and Algorithms, and $2)$ `ICP` can in fact improve accuracy of baseline algorithms. We conduct a perturbation analysis to show that `ICP` framework error bounds are not affected by the perturbations, rather the efficiency is affected.

The efficiency of `ICP` predictions are affected by the conformity function we use. In this work, we consider the probability margin nonconformity measure. It would be interesting to explore other conformity measures specific to graph data in the future. Another future direction would be to adopt `ICP` framework to other network prediction problems such as link prediction.

CHAPTER 6 CONCLUSION

Social network analysis provides a convenient platform to analyze and understand interactions among social entities. Social networks are widely studied to find key players/influential nodes in a network. These works find people of interest (POIs), who are important in terms of the network structure. There are many applications such as security (finding criminals in networks), marketing (targeted advertising), friend recommendation that would benefit from identifying POIs with certain properties that do not necessarily correlate with the network structure. We propose algorithms to identify such POIs in this dissertation.

First, in Chapter 2, we discuss the key player identification problem but in a heterogeneous, dynamic network setting to predict future key players (rising stars) in social networks. We propose `MOO` algorithm to find rising stars, which first create homogeneous networks from heterogeneous data to find potential rising from each network. We propose multi-objective optimization and rank aggregation approaches to combine potential rising stars from various networks to predict the final set of rising stars. We apply `MOO` algorithm to academic data and question answer forum data to show that, `MOO` algorithm predict rising stars who truly become key players in the future. Further, we show that the predicted rising stars out perform state of the art rising star prediction algorithms in academic data.

The second problem we consider is collecting network data to identify as many POIs possible in a social network. We specifically consider a dark network setting, where POIs are adversarial towards a data collector to hide their identity. In Chapter 3, we propose `RedLearn` and `RedLearnRS` algorithms to locate POIs in dark networks. `RedLearn` algorithm assumes that POIs report their true status when queried while `RedLearnRS` algorithm dispenses that assumption. We use real world terrorist networks, social networks and synthetic terrorist networks to show that `RedLearn` and `RedLearnRS` perform well to identify POIs through network crawling while being robust to various types of errors dark network data collection can incur.

Even though we discuss various errors network data collection can face in an adversarial setting in Chapter 3, there are no realistic models to explain such behavior. In Chapter 4, we propose Adversarial Social Network Analysis game (`ASNA game`) framework to study interactions between a data collector and members of an adversarial group. `ASNA game` is a single player network game, where a participant plays as a member of the adversarial group and report to a data collector whether their neighbors in the network are adversarial or not. We conduct experiments with recruited participants from Amazon mechanical Turk to show that players understand key aspects of the game and act as part of the adversarial group. We intend to extend the game to a multi player setting where we would allow participants to communicate with each other. We also intend to use the existing game framework to understand other types of adversarial data collection problems.

In Chapter 5, we consider POI identification problem as a semi-supervised learning problem, where POIs are partially labeled in a network. We seek to identify rest of the POIs by modeling this problem as a classification problem. There are a multitude of node classification algorithms which are specifically designed to conduct classification in network data. In applications where POI prediction is a sensitive matter, such as predicting an individual as a criminal or not a criminal, a user may want to associate a confidence measure with predictions to obtain guaranteed error bounds. In this chapter, we show how to apply conformal prediction framework to obtain bounded error rates for node classification problem. Experimental analysis shows that, 1) The error rates are close to expected errors, 2) Conformal prediction framework provides higher prediction accuracy than the original predictions, showing that we are in fact correcting some of the errors made by the original node classification algorithms.

We investigate locating POIs in social networks through multiple facets in this dissertation. We propose algorithms to find, collect data to identify, model and predict POIs in social networks.

Bibliography

[1] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," Physical review E, vol. 64, no. 4, p. 046135, 2001.

[2] D. Aldous and J. Fill, "Reversible markov chains and random walks on graphs," 2002.

[3] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06). IEEE, 2006, pp. 475–486.

[4] A. Asztalos and Z. Toroczkai, "Network discovery by generalized random walks," EPL (Europhysics Letters), vol. 92, no. 5, p. 50008, 2010.

[5] K. Avrachenkov, P. Basu, G. Neglia, B. Ribeiro, and D. Towsley, "Pay few, influence most: Online myopic network covering," in IEEE NetSciCom Workshop, 2014.

[6] W. E. Baker and R. R. Faulkner, "The social organization of conspiracy: Illegal networks in the heavy electrical equipment industry," American sociological review, pp. 837–860, 1993.

[7] L. Benkherouf and J. Bather, "Oil exploration: sequential decisions in the face of uncertainty," Journal of Applied Probability, vol. 25, no. 3, pp. 529–543, 1988.

[8] S. Bhagat, G. Cormode, and S. Muthukrishnan, “Node classification in social networks,” in Social network data analytics. Springer, 2011, pp. 115–148.

[9] P. Biernacki and D. Waldorf, “Snowball sampling: Problems and techniques of chain referral sampling,” Soc. methods & research, vol. 10, no. 2, pp. 141–163, 1981.

[10] C. A. Bliss, C. M. Danforth, and P. S. Dodds, “Estimation of global network statistics from incomplete data,” PloS one, vol. 9, no. 10, p. e108471, 2014.

[11] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, “Social network search as a volatile multi-armed bandit problem,” HUMAN, vol. 2, no. 2, p. 84, 2013.

[12] P. Bonacich, “Some unique properties of eigenvector centrality,” Social networks, vol. 29, no. 4, pp. 555–564, 2007.

[13] S. P. Borgatti, “Identifying sets of key players in a social network,” Computational & Mathematical Organization Theory, vol. 12, no. 1, pp. 21–34, 2006.

[14] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, “Network analysis in the social sciences,” science, vol. 323, no. 5916, pp. 892–895, 2009.

[15] U. Brandes and D. Fleischer, “Centrality measures based on current flow,” in Annual symposium on theoretical aspects of computer science. Springer, 2005, pp. 533–544.

[16] M. Braverman, O. Etesami, E. Mossel et al., “Mafia: A theoretical study of players and coalitions in a partial information environment,” The Annals of Applied Probability, vol. 18, no. 3, pp. 825–846, 2008.

[17] C. Burfoot, S. Bird, and T. Baldwin, “Collective classification of congressional floor-debate transcripts,” in Proceedings of the 49th Annual Meeting

of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011, pp. 1506–1515.

[18] D. Cai, Z. Shao, X. He, X. Yan, and J. Han, “Mining hidden community in heterogeneous social networks,” in Proceedings of the 3rd international workshop on Link discovery. ACM, 2005, pp. 58–65.

[19] V. R. Carvalho and W. W. Cohen, “On the collective classification of email speech acts,” in Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2005, pp. 345–352.

[20] H. Chen, W. Chung, J. J. Xu, G. Wang, Y. Qin, and M. Chau, “Crime data mining: a general framework and some examples,” computer, vol. 37, no. 4, pp. 50–56, 2004.

[21] M. Dashevskiy and Z. Luo, “Time series prediction with performance guarantee,” IET communications, vol. 5, no. 8, pp. 1044–1051, 2011.

[22] A. Daud, R. Abbasi, and F. Muhammad, “Finding rising stars in social networks,” in Database Systems for Advanced Applications. Springer, 2013, pp. 13–24.

[23] A. Daud, M. Ahmad, M. Malik, and D. Che, “Using machine learning techniques for rising star prediction in co-author network,” Scientometrics, pp. 1–25, 2014.

[24] B. Davis, R. Gera, G. Lazzaro, B. Y. Lim, and E. C. Rye, “The marginal benefit of monitor placement on networks,” in Complex Networks VII. Springer, 2016, pp. 93–104.

[25] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Generalized louvain method for community detection in large networks," in 2011 11th International Conference on Intelligent Systems Design and Applications. IEEE, 2011, pp. 88–93.

[26] D. Devetyarov and I. Nouretdinov, "Prediction with confidence based on a random forest classifier," in IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, 2010, pp. 37–44.

[27] L. Ding, S. Zeng, and L. Kang, "A fast algorithm on finding the non-dominated set in multi-objective optimization," in Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, vol. 4. IEEE, 2003, pp. 2565–2571.

[28] Y. Ding, E. Yan, A. Frazho, and J. Caverlee, "Pagerank for ranking authors in co-citation networks," Journal of the American Society for Information Science and Technology, vol. 60, no. 11, pp. 2229–2243, 2009.

[29] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, "K-core organization of complex networks," Physical review letters, vol. 96, no. 4, p. 040601, 2006.

[30] P. Erdos and A. Rényi, "On the evolution of random graphs," Publ. Math. Inst. Hung. Acad. Sci, vol. 5, no. 1, pp. 17–60, 1960.

[31] K. Fitch and N. E. Leonard, "Information centrality and optimal leader selection in noisy networks," in 52nd IEEE Conference on Decision and Control. IEEE, 2013, pp. 7510–7515.

[32] M. Franceschet, "A comparison of bibliometric indicators for computer science scholars and journals on web of science and google scholar," Scientometrics, vol. 83, no. 1, pp. 243–258, 2010.

[33] L. C. Freeman, "A set of measures of centrality based on betweenness," Sociometry, pp. 35–41, 1977.

[34] N. Friedman and D. Koller, "Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks," Machine learning, vol. 50, no. 1-2, pp. 95–125, 2003.

[35] A. Fronczak and P. Fronczak, "Biased random walks in complex networks: The role of local navigation rules," Physical Review E, vol. 80, no. 1, p. 016107, 2009.

[36] M. Galeotti, Global crime today: the changing face of organised crime. Routledge, 2014.

[37] B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos, "Using ghost edges for classification in sparsely labeled networks," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008, pp. 256–264.

[38] R. Gera, R. Miller, M. MirandaLopez, S. Warnke, and A. Saxena, "Three is the answer: Combining relationships to analyze multilayered terrorist networks," in Advances in Social Networks Analysis and Mining (ASONAM), 2017 IEEE/ACM. IEEE, 2017.

[39] M. Girvan and M. E. Newman, "Community structure in social and biological networks," Proceedings of the national academy of sciences, vol. 99, no. 12, pp. 7821–7826, 2002.

[40] M. S. Granovetter, "The strength of weak ties," in Social networks. Elsevier, 1977, pp. 347–367.

[41] S. Gregory, "Finding overlapping communities in networks by label propagation," New Journal of Physics, vol. 12, no. 10, p. 103018, 2010.

[42] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016, pp. 855–864.

[43] S. Hanneke and E. P. Xing, "Network completing and survey sampling," in AISTATS, 2009, pp. 209–215.

[44] M. Hartwig, P. Anders Granhag, and L. A. Strömwall, "Guilty and innocent suspects' strategies during police interrogations," Psychology, Crime & Law, vol. 13, no. 2, 2007.

[45] M. Hartwig, P. A. Granhag, L. A. Strömwall, and A. Vrij, "Detecting deception via strategic disclosure of evidence," Law and human behavior, vol. 29, no. 4, pp. 469–484, 2005.

[46] D. Haussler, Probably approximately correct learning. University of California, Santa Cruz, Computer Research Laboratory, 1990.

[47] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," Physical review E, vol. 65, no. 2, p. 026107, 2002.

[48] X. Huang, H. Cheng, and J. X. Yu, "Dense community detection in multi-valued attributed networks," Information Sciences, vol. 314, pp. 77–99, 2015.

[49] B. D. Hughes, "Random walks and random environments," Oxford, vol. 2, pp. 1995–1996, 1995.

[50] S. E. Inc. (2016) Information security stackexchange. [Online]. Available: http://security.stackexchange.com/

[51] U. Johansson, H. Boström, and T. Löfström, "Conformal prediction using decision trees," in 2013 IEEE 13th international conference on data mining, 2013.

[52] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," Physical review E, vol. 83, no. 1, p. 016107, 2011.

[53] L. Katz, "A new status index derived from sociometric analysis," Psychometrika, vol. 18, no. 1, pp. 39–43, 1953.

[54] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.

[55] A. S. Klovdahl, "Social networks and the spread of infectious diseases: the aids example," Social science & medicine, vol. 21, no. 11, pp. 1203–1216, 1985.

[56] S. Koschade, "A social network analysis of jemaah islamiyah: The applications to counterterrorism and intelligence," Studies in Conflict & Terrorism, vol. 29, no. 6, pp. 559–575, 2006.

[57] V. Le, "Organised crime typologies: Structure, activities and conditions," International Journal of Criminology and Sociology, vol. 1, pp. 121–131, 2012.

[58] J. B. Lee and H. Adorna, "Link prediction in a modified heterogeneous bibliographic network," in Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012). IEEE Computer Society, 2012, pp. 442–449.

[59] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006, pp. 631–636.

[60] C.-T. Li and S.-D. Lin, "Centrality analysis, role-based clustering, and egocentric abstraction for heterogeneous social networks," in Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom). IEEE, 2012, pp. 1–10.

[61] X.-L. Li, C. S. Foo, K. L. Tew, and S.-K. Ng, "Searching for rising stars in bibliography networks," in Database Systems for Advanced Applications. Springer, 2009, pp. 288–292.

[62] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: an embedding approach," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[63] F. Lin and W. W. Cohen, "Semi-supervised classification of network data using very few labels," in Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on. IEEE, 2010, pp. 192–199.

[64] X. Liu, J. Bollen, M. L. Nelson, and H. Van de Sompel, "Co-authorship networks in the digital library research community," Information processing & management, vol. 41, no. 6, pp. 1462–1480, 2005.

[65] Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li, "Supervised rank aggregation," in Proceedings of the 16th international conference on World Wide Web. ACM, 2007, pp. 481–490.

[66] Q. Lu and L. Getoor, "Link-based classification," in Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 496–503.

[67] Y. Lu, X. Luo, M. Polgar, and Y. Cao, "Social network analysis of a criminal hacker community," Journal of Computer Information Systems, vol. 51, no. 2, pp. 31–41, 2010.

[68] S. A. Macskassy and F. Provost, "Suspicion scoring based on guilt-by-association, collective inference, and focused data access," in International conference on intelligence analysis, 2005.

[69] A. S. Maiya and T. Y. Berger-Wolf, "Online sampling of high centrality individuals in social networks," in PAKDD, 2010, pp. 91–98.

[70] L. Makili, J. Vega, S. Dormido-Canto, I. Pastor, and A. Murari, "Computationally efficient svm multi-class image recognition with confidence measures," Fusion Engineering and Design, vol. 86, no. 6-8, pp. 1213–1216, 2011.

[71] W. Mason and S. Suri, "Conducting behavioral research on amazon's mechanical turk," Behavior research methods, vol. 44, no. 1, pp. 1–23, 2012.

[72] L. I. Meho, "The rise and rise of citation analysis," arXiv preprint physics/0701012, 2006.

[73] T. P. Michalak, T. Rahwan, and M. Wooldridge, "Strategic social network analysis." in AAAI, 2017, pp. 4841–4845.

[74] J. Motl and O. Schulte, "The ctu prague relational learning repository," arXiv preprint arXiv:1511.03086, 2015.

[75] G. Namata, B. London, L. Getoor, B. Huang, and U. EDU, "Query-driven active surveying for collective classification," in 10th International Workshop on Mining and Learning with Graphs, 2012, p. 8.

[76] J. Neville and D. Jensen, "Iterative classification in relational data," in Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data, 2000, pp. 13–20.

[77] J. D. Noh and H. Rieger, "Random walks on complex networks," Physical review letters, vol. 92, no. 11, p. 118701, 2004.

[78] A. A. Novikov and A. N. Shiryaev, "On an effective solution of the optimal stopping problem for random walks," Theory of Probability & Its Applications, vol. 49, no. 2, pp. 344–354, 2005.

[79] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[80] G. Paolacci, J. Chandler, and P. G. Ipeirotis, "Running experiments on amazon mechanical turk," Judgment and Decision making, vol. 5, no. 5, pp. 411–419, 2010.

[81] H. Papadopoulos, "Inductive conformal prediction: Theory and application to neural networks," in Tools in artificial intelligence. IntechOpen, 2008.

[82] H. Papadopoulos, V. Vovk, and A. Gammerman, "Regression conformal prediction with nearest neighbours," Journal of Artificial Intelligence Research, 2011.

[83] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014, pp. 701–710.

[84] J. Raab and H. B. Milward, "Dark networks as problems," Journal of public administration research and theory, vol. 13, no. 4, pp. 413–439, 2003.

[85] A. H. Rasti, M. Torkjazi, R. Rejaie, N. Duffield, W. Willinger, and D. Stutzbach, “Respondent-driven sampling for characterizing unstructured overlays,” in IEEE INFOCOM 2009. IEEE, 2009, pp. 2701–2705.

[86] N. Roberts and S. Everton, “Terrorist data: Noordin top terrorist network,” https://sites.google.com/site/sfeverton18/research/appendix-1, Jun. 2011.

[87] M. Rosvall and C. T. Bergstrom, “Maps of random walks on complex networks reveal community structure,” Proceedings of the National Academy of Sciences, vol. 105, no. 4, pp. 1118–1123, 2008.

[88] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee et al., “Boosting the margin: A new explanation for the effectiveness of voting methods,” The annals of statistics, vol. 26, no. 5, pp. 1651–1686, 1998.

[89] D. M. Schwartz and T. D. Rouselle, “Using social network analysis to target criminal networks,” Trends in Organized Crime, vol. 12, no. 2, pp. 188–207, 2009.

[90] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” AI magazine, vol. 29, no. 3, pp. 93–93, 2008.

[91] G. Shafer and V. Vovk, “A tutorial on conformal prediction,” Journal of Machine Learning Research, vol. 9, no. Mar, pp. 371–421, 2008.

[92] M. K. Sparrow, “The application of network analysis to criminal intelligence: An assessment of the prospects,” Social networks, vol. 13, no. 3, pp. 251–274, 1991.

[93] R. T. Stern, L. Samama, R. Puzis, T. Beja, Z. Bnaya, and A. Felner, "Tonic: Target oriented network intelligence collection for the social web." in AAAI, 2013.

[94] L. A. Strömwall and R. M. Willén, "Inside criminal minds: Offenders' strategies when lying," Journal of Investigative Psychology and Offender Profiling, vol. 8, no. 3, 2011.

[95] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," IEEE/ACM Transactions on Networking (TON), vol. 17, no. 2, pp. 377–390, 2009.

[96] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla, "When will it happen?: relationship prediction in heterogeneous information networks," in Proceedings of the fifth ACM international conference on Web search and data mining. ACM, 2012, pp. 663–672.

[97] S. Suri and D. J. Watts, "Cooperation and contagion in web-based, networked public goods experiments," PloS one, vol. 6, no. 3, p. e16836, 2011.

[98] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in Proceedings of the 24th international conference on world wide web. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[99] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008, pp. 990–998.

[100] L. Tang, X. Wang, and H. Liu, "Uncoverning groups via heterogeneous interaction analysis," in Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on. IEEE, 2009, pp. 503–512.

[101] G. Tsatsaronis, I. Varlamis, S. Torge, M. Reimann, K. Nørvåg, M. Schroeder, and M. Zschunke, "How to become a group leader? or modeling author types based on graph mining," in Research and Advanced Technology for Digital Libraries. Springer, 2011, pp. 15–26.

[102] J. N. Tsitsiklis and B. Van Roy, "Optimal stopping of markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives," IEEE Transactions on Automatic Control, vol. 44, no. 10, pp. 1840–1851, 1999.

[103] V. Vovk, A. Gammerman, and G. Shafer, Algorithmic learning in a random world. Springer Science & Business Media, 2005.

[104] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," nature, vol. 393, no. 6684, p. 440, 1998.

[105] P. Wijegunawardana, R. Gera, and S. Soundarajan, "Node classification with bounded error rates," in Conference on Complex Networks CompleNet. Springer, 2020.

[106] P. Wijegunawardana, K. Mehrotra, and C. Mohan, "Finding rising stars in heterogeneous social networks," in 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2016, pp. 614–618.

[107] P. Wijegunawardana, V. Ojha, R. Gera, and S. Soundarajan, "Seeing red: Locating people of interest in networks," in Conference on Complex Networks CompleNet. Springer, 2017, pp. 141–150.

[108] ——, "Sampling dark networks to locate people of interest," Social Network Analysis and Mining, vol. 8, no. 1, 2018.

[109] P. Wijegunawardana, M. Taglic, R. Gera, and S. Soundarajan, "An experimental framework for characterizing adversarial behavior in social networks," in SBP-BRiMS, 2019.

[110] W. H. Wong and P. A. Brown, "E-bandits in global activism: Wikileaks, anonymous, and the politics of no one," Perspectives on Politics, vol. 11, no. 4, pp. 1015–1033, 2013.

[111] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in Proceedings of the 19th international conference on World wide web. ACM, 2010, pp. 981–990.

[112] J. Xu and H. Chen, "The topology of dark networks," Communications of the ACM, vol. 51, no. 10, pp. 58–65, 2008.

[113] G. Yan, "Peri-watchdog: Hunting for hidden botnets in the periphery of online social networks," Computer Networks, vol. 57, no. 2, pp. 540–555, 2013.

[114] R. Zafarani and H. Liu, "Social computing data repository at ASU," 2009. [Online]. Available: http://socialcomputing.asu.edu

[115] E. Zheleva and L. Getoor, "To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles," in Proceedings of the 18th international conference on World wide web. ACM, 2009, pp. 531–540.

[116] D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles, "Co-ranking authors and documents in a heterogeneous network," in Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on. IEEE, 2007, pp. 739–744.

[117] X. Zhu, Z. Ghahramani, J. Lafferty et al., “Semi-supervised learning using gaussian fields and harmonic functions,” in ICML, vol. 3, 2003, pp. 912–919.

Vita

NAME OF AUTHOR: Pivithuru Wijegunawardana

PLACE OF BIRTH: Colombo, Sri Lanka

DATE OF BIRTH: June 6^{th}, 1987

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

Syracuse University, New York, USA

University of Moratuwa, Sri Lanka

DEGREES AWARDED:

M.S Computer Science, 2019, Syracuse University, USA

B.Sc (Hons) Computer Science and Engineering, 2011, University of Moratuwa, Sri Lanka

PUBLICATIONS:

- Pivithuru Wijegunawardana, Ralucca Gera, and Sucheta Soundarajan. Node Classification with Bounded Error Rates, CompleNet, March, 2020 (forthcoming).

- Pivithuru Wijegunawardana, Mario Taglic, Ralucca Gera, and Sucheta Soundarajan. An Experimental Framework for Characterizing Adversarial Behavior in Social Networks, SBP-BRiMS, July, 2019.
- Pivithuru Wijegunawardana, Vatsal Ojha, Ralucca Gera, and Sucheta Soundarajan. Sampling dark networks to locate people of interest, SNAM, 2018.
- Pivithuru Wijegunawardana, Vatsal Ojha, Ralucca Gera, and Sucheta Soundarajan. Seeing Red: Locating People of Interest in Networks, ComPleNet, March 2017.
- Pivithuru Wijegunawardana, Kishan Mehrotra, Chilukuri Mohan. Finding Rising Stars in Social Networks, ICTAI, November 2016.

PROFESSIONAL EXPERIENCE:

- Intern Data Scientist- Conversant LLC,Chicago, IL (May 2016 - Aug 2016)
- Lecturer - University of Moratuwa, Sri Lanka (Dec 2011 - Aug 2013)