Impact of Social Structure on Wireless Networking:

Modeling and Utility

by

Brian Proulx

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved April 2015 by the
Graduate Supervisory Committee:

Junshan Zhang, Chair
Douglas Cochran
Lei Ying
Yanchao Zhang

ARIZONA STATE UNIVERSITY

May 2015

ABSTRACT

The explosive growth of data generated from different services has opened a new vein of research commonly called “big data.” The sheer volume of the information in this data has yielded new applications in a wide range of fields, but the difficulties inherent in processing the enormous amount of data, as well as the rate at which it is generated, also give rise to significant challenges. In particular, processing, modeling, and understanding the structure of online social networks is computationally difficult due to these challenges. The goal of this study is twofold: first to present a new networked data processing framework to model this social structure, and second to highlight the wireless networking gains possible by using this social structure.

The first part of the dissertation considers a new method for modeling social networks via probabilistic graphical models. Specifically, this new method employs the t-cherry junction tree, a recent advancement in probabilistic graphical models, to develop a compact representation and good approximation of an otherwise intractable probabilistic model. There are a number of advantages in this approach: 1) the best approximation possible via junction trees belongs to the class of t-cherry junction trees; 2) constructing a t-cherry junction tree can be largely parallelized; and 3) inference can be performed using distributed computation. To improve the quality of approximation, an algorithm to build a higher order tree gracefully from an existing one, without constructing it from scratch, is developed. this approach is applied to Twitter data containing 100,000 nodes to study the problem of recommending connections to new users.

Next, the t-cherry junction tree framework is extended by considering the impact of estimating the distributions involved from a training data set. Understanding this impact is vital to real-world applications as distributions are not known perfectly, but rather generated from training data. First, the fidelity of the t-cherry junction tree approximation due to this estimation is quantified. Then the scaling behavior, in terms of the size of the t-cherry junction tree, is approximated to show that higher-order t-cherry junction trees, which with perfect information are higher fidelity approximations, may actually result in decreased fidelity due to the difficulties in accurately estimating higher-dimensional distributions. Finally, this part concludes by demonstrating these findings by considering a distributed detection situation in which the sensors' measurements are correlated.

Having developed a framework to model social structure in online social networks, the study then highlights two approaches for utilizing this social network data in existing wireless communication networks. The first approach is a novel application: using social networks to enhance device-to-device wireless communication. It is well known that wireless communication can be significantly improved by utilizing relays to aid in transmission. Rather than deploying dedicated relays, a system is designed in which users can relay traffic for other users if there is a shared social trust between them, e.g., they are "friends" on Facebook, and for users that do not share social trust, implements a coalitional game framework to motivate users to relay traffic for each other. This framework guarantees that all users improve their throughput via relaying while ensuring that each user will function as a relay only if there is a social trust relationship or, if there is no social trust, a cycle of reciprocity is established in which a set of users will agree to relay for each other. This new system shows significant throughput gain in simulated networks that utilize real-world social network traces.

The second application of social structure to wireless communication is an approach to reduce the congestion in cellular networks during peak times. This is achieved by two means: preloading and offloading. Preloading refers to the process of using social network data to predict user demand and serve some users early, before the cellular network traffic peaks. Offloading allows users that have already obtained a copy of the content to opportunistically serve other users using device-to-device communication, thus eliminating the need for some cellular traffic. These two methods work especially well in tandem, as preloading creates a base of users that can serve later users via offloading. These two processes can greatly reduce the peak cellular traffic under ideal conditions, and in a more realistic situation, the impact of uncertainty in human mobility and the social network structure is analyzed. Even with the randomness inherent in these processes, both preloading and offloading offer substantial improvement. Finally, potential difficulties in preloading multiple pieces of content simultaneously are highlighted, and a heuristic method to solve these challenges is developed.

ACKNOWLEDGEMENTS

I would like to express my thanks to my advisor, Dr. Junshan Zhang, for his support and encouragement. I am very lucky to have him as my advisor.

Also, I would like to thank Dr. Douglas Cochran for all of his help in both my research and professional development.

My fellow research group members were invaluable during my time here at ASU.

Finally, I would like to acknowledge my wife Kelly, without whom I never would have achieved this, as well as the rest of my family.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Overview

In the past decade, the explosive growth of online social networks has transformed interactions between people. Remaining in touch with distant family, or even brief acquaintances, is easier than ever, especially with the rise of smartphones. These allow people to be connected to online social networks at essentially all times of day. Understanding the social structure in online social networks can create new avenues of research that develop applications to leverage this information. However, the scale of social networks is enormous, which poses many challenging technical problems. Storing and processing the massive amount of data needed to analyze social networks requires new techniques that consider the scale of the problem first.

In a broader sense, modeling the social tie structure in online social networks is a part of the research field called "big data." This field studies problems in processing and understanding very large data sets, typically specified by the "3 Vs": volume: the size of the data set, velocity: the rate at which the data is generated, and variety: the wide range of data types. The field of big data can be coarsely divided into two parts, namely developing techniques to process the data and new applications that leverage this processing. The challenges of processing the data seen in typical big data applications are numerous, and new approaches and computing platforms have been devised to solve some of these issues. After these are used to process the data into a useable form, new applications can be developed that utilize this incredible amount of information contained in the data set.

Specifically, this dissertation focuses on both of these fundamental problems. The first problem considered in this dissertation is a new technique for processing online social network data in order to model the social network structure. However, as social networks are of massive size, the largest being on the scale of the population of Earth, an approach that can work at this scale is needed. One fundamental model of this problem is developed in [1], where different graph-theoretic tools were used. Also, factor graph models [2] and feature-based approaches [3] have been proposed. However, these approaches do not perform well for large-scale networks, especially if feature data is used, as the amount of feature data for each node and link is prohibitive. However, probabilistic graphical models is a set of tools specifically developed to model probability distributions at these massive scales [4]. In particular, the t-cherry junction tree [5] can handle large data sets and has a quantitative performance metric, unlike other probabilistic graphical models such as the Bayesian network [6]. The t-cherry junction tree operates by considering small subsets of random variables connected together in a tree structure to create an approximation to the high-dimensional joint distribution.

Another important aspect is the performance of the t-cherry junction tree when distributions are estimated from data, rather than being perfectly known. This models the real-world utilization of this tool, as most problems consider a set of training data to build the model. The fidelity of the approximation depends upon the ability to accurately estimate the marginal distributions. The scaling behavior, both in terms of the size of the training data set and the order of the t-cherry junction tree, is approximated to highlight the fact that capturing higher-order dependence structures (which with perfect information results in a higher fidelity) can result in a loss of fidelity if there is inadequate training data for estimating the distributions. This tradeoff is quantified so that the proper order of the junction tree can be determined.

The second problem studied in this dissertation is how to utilize the social structure that is modeled in the first part to improve different aspects of wireless communication. Relaying, in which one wireless user helps another by rebroadcasting their information, is known to greatly increase throughput [7]. However, as relaying for another user consumes resources, users are generally not willing to relay traffic without incentive. Previous studies on incentivizing relaying have focused on payment methods [8] or reputation systems [9], in which all users' histories of relaying is tracked. A new approach to stimulate relaying is to consider using existing social ties between users to allow them to relay for those people they know and are willing to help for no reward. In addition to this insight, users may still be incentivized to relay for other users without existing social ties if a cycle of reciprocity is developed. That is, a user may relay for another user if that user relays for them, or a larger cycle of reciprocity may be constructed. This process allows the throughput gains of relaying to be realized without the assumption that every node acts purely altruistically. Again, this highlights the strength of leveraging existing social network data in traditional communication problems.

Another application of the modeled social network data is to alleviate cellular traffic overloading. Popular online content can become viral, that is, the number of people who want to consume the content grows very quickly, and serving this viral demand over a cellular network is difficult as cellular data networks are becoming increasingly overloaded. Moving this traffic earlier in time, preloading, has been proposed as this flattens the demand curve. However, the specifics of which user to move to which time and the exact nature of the future cellular traffic are ongoing challenges. Previous solutions considered *a priori* knowledge of future cellular traffic or simple Markov chain monte carlo approaches for predicting the future traffic [10]. Another approach uses the Bass diffusion model to approximate the demand curve and preload users accordingly [11]. The advent of online social networks allows a cellular provider to leverage this information about the social structure to predict the future cellular traffic when viral content is spread over these social networks. This key insight allows the cellular network to better predict, and thus better distribute, the traffic load over time to ensure that timely service of the content is achieved.

1.2 Summary of Main Contributions

This dissertation focuses on the problems associated with modeling social ties at scale in Chapter 2 and the performance of this model using a data driven approach in Chapter 3. Next, the potential gains available when using existing social network structure in wireless networks in Chapters 5 and 4 are investigated.

1.3 Modeling Social Network Relationships

In Chapter 2, the problems inherent in modeling social network relationships are considered. The enormous scale of social networks makes data mining and modeling techniques difficult. Algorithms constructed to model different aspects of social networks must either have limited complexity or must be offloaded to cloud computing or similar services to ensure that a reasonable run time is achieved. In order to trade off between modeling accuracy and run time, we incorporate a framework from probabilistic graphical models: the t-cherry junction tree.

The core modeling problem that is addressed in Chapter 2 is approximating multivariate joint probability distributions that have a large number of random variables. The number of entries required to store these distributions grows exponentially in the number of random variables involved. Thus approximating using a number of low-order marginal distributions can greatly reduce the storage size of the joint distribution. Using t-cherry junction trees, approximations to the true joint distribution can be constructed from small marginal distributions while being able to explicitly calculate the Kullback-Leibler divergence between the approximation and the true joint distribution, as opposed to heuristic methods. Additionally, t-cherry junction trees can used to perform exact polynomial time inference.

We present a greedy algorithm to construct a t-cherry junction tree of a given order, meaning a fixed number of random variables in each marginal distribution, in order to approximate a joint distribution. Constructing the t-cherry junction tree representing the best approximation, the lowest Kullback-Leibler divergence between the approximation and the true distribution, is an NP-hard problem, so the greedy algorithm does not guarantee the best approximation. However, even this greedy algorithm requires a long time to run. Another strength of the t-cherry junction tree is that if a better approximation is desired, the order of the tree can be increased. We present a method to increase the order of a given tree without constructing a t-cherry junction tree from scratch, which reduces the time necessary by orders of magnitude, though there is no performance guarantee involved in this method.

At the end of this chapter is an example application of this new approach. A Twitter data set containing 100,000 users is considered as training data to recommend existing users that a user new to Twitter might be interested in following. This problem is formulated as a joint distribution containing 100,000 binary random variables that is then approximated using a 4-order t-cherry junction tree. After partitioning and rejoining the junction tree, an approximate joint distribution is generated that requires only 2,399,920 data entries, as opposed to $2^{100,000}$ for the true distribution, and the Kullback-Leibler divergence between this approximation and the true distribution can be exactly calculated as 13,511.

1.4 Impact of the Data Driven Approach

In many real-world applications, probability distributions are estimated using a training data set. As the training data set is a fixed size, the estimated distributions are not exactly equal to the true distributions. In Chapter 3, the impact of using estimated distributions is explicitly, quantitatively stated in terms of the Kullback-Leibler divergence. The divergence retains the same decomposition as the case when the distributions are known exactly, with an additional term for each cluster and separator in the tree.

This divergence result allows for many conclusions about the change in behavior to the t-cherry junction tree in this case. The first is that these additional terms are non-negative, meaning that using estimated distributions always results in a worse approximation to the true high-dimensional distribution. The exact behavior of these new terms depends on the exact form of the training data set, but the scaling behavior of the expected value is determined. The scaling behavior behaves as the reciprocal of the number of samples in the training data set, i.e., more training data results in improved performance. Also, as the order of the junction tree, the number of random variables in each cluster, is increased, the loss in fidelity scales exponentially in the order. These two observations imply that for a range of amounts of training data, the best overall fidelity is achieved not at the highest order, as when the distributions are exactly known, but instead at an order that trades off more complex dependency structures against the decreased accuracy in estimating higher order distributions.

These concepts are demonstrated with a sample application at the end of this chapter. The problem of distributed detection is considered, in which many sensors take measurements and individually decide whether a target is present or absent. However, these sensor measurements have a certain correlation structure, and thus their decisions are correlated. These correlated decisions are aggregated at a data fusion center, which performs a likelihood ratio test and arrives at a global decision. This test can be approximated using t-cherry junction trees to account for some correlation structure while not requiring massive storage and estimation penalties. Low-order trees are shown to retain most of the theoretical performance, even when they are estimated from training data.

1.5 Enabling Cooperative Relaying Via Social Tie Structure

We present an application of social networks in Chapter 4. In this chapter, the challenge of ensuring fair cooperative communication is addressed. The key insight is that mobile devices are carried by humans who have relationships with each other. The interplay of the physical and social graphs of these users is explored to improve the throughput of device-to-device communication.

This is essentially a relay selection problem, but without any dedicated relays. Each user can calculate the increased throughput when using any other user as a relay, and each user wants to select the user that offers the highest throughput. If there is social trust between these users (they are connected in the online social network) then it is assumed that these users will offer to relay for each other without any personal benefit. However, if there is no social relationship with the desired relay, a relaying agreement can be reached, but only if each participant receives a benefit. That is, no user will relay an unknown user unless there is a cycle of reciprocity created which allows each user to select a favorable relay while agreeing to relay for another user. For example, user A will relay for user B who will relay for user C, who in turn relays for user A. This creates a relaying cycle A-B-C-A.

This problem is formulated as a coalitional game. We demonstrate that this game has the top coalition property, which ensures that the game has a core solution. A centralized algorithm is developed to return this core solution that is immune to group deviations, individually rational, truthful, and computationally efficient. The performance of this algorithm is analyzed by simulating a two-dimensional arrangement of users and assigning those users identities in a social network. We use both Erdos-Renyi random graphs and real social network data sets to show that the overall throughput of the system can be improved by over 100%.

1.6 Reducing Peak Cellular Traffic

In Chapter 5, we present a method to reduce the peak traffic of cellular base stations by utilizing online social networks. As a whole, the cellular network in the United States is operating near capacity during busy periods. This is a growing concern as the amount of cellular data traffic has risen at an exponential rate over the past several years. During the busiest parts of the day, it is difficult for cellular base stations to serve all of their users in a timely manner.

We focus on cellular traffic that is generated by interest diffusion in online social networks. This focus is due to the fact that some content on a social network can become viral, meaning that many users in a social network will download that content in a short period of time, which implies that this traffic can be predicted beforehand. We propose a method to reduce these large spikes in traffic based upon viral content.

This method is comprised of two parts, preloading and offloading. Preloading content is serving users before they become interested in the content and request it. For instance, if it can be predicted that a user will become interested in the content later, that user can be served before the large spike in traffic, when the base station is less congested. Offloading is based on the notion that cellular users are generally mobile, and thus device-to-device (D2D) communication can be used. If a user is interested in a piece of content, it is possible to deliver the content from another user that has already downloaded the content using D2D communication. In this way, the cellular base station is not required to serve the user.

In order to quantify the performance of these two approaches, we focus on the impact of uncertainty in the interest diffusion and content delivery processes. The interest diffusion process, in which users become interested in the content via sharing over the OSN is random as the exact structure and sharing behavior of the social network is not known *a priori*. The second process is the content delivery process, by which users are served by either the cellular network or D2D communication. The uncertainty in this process is due to human mobility: it is inherently random whether a user will come into the physical proximity of another user that already has a copy of the content.

We develop a greedy algorithm that obtains minimal value of the maximum cellular load when both of these processes are known *a priori*. From this, it is shown that both preloading and offloading offer vastly improved performance. The random nature of human mobility is also shown to have only a small effect on the reduction in peak traffic, while the random structure of the social network can cause a significant decline in the ability to decrease cellular peak load. Lastly, we extend this framework to multiple pieces of content to highlight the additionaly difficulties present and propose a heuristic solution.

1.7 Related Work

Many studies have examined similar problems with different focuses. Modeling social tie structure has been studied using a wide variety of techniques. Originally, the problem was approached by utilizing different graph theory metrics that only rely on the structure of the network [1]. However, this requires a full model of the social network, which is impractical in modern social networks. Other probabilistic graphical model tools have been used, namely factor graphs [2], but this tool has no qualitative performance metric, and in fact, may not converge at all [12]. Using outside information, such as location data, has been studied to further increase the accuracy of the prediction [3]. But again, this approach does not scale to the massive amount of data contained in real-world social networks.

In Chapter 3, a data-driven approach is developed that utilizes only training data to approximate the dependence structure. Other works have studied directly learning the underlying Markov random field [13]. However, this only attempts to learn the underlying dependence structure, not approximate it, so the resulting complexity will not be reduced. Additionally, the primary focus was an upper bound on the number of samples required for accurate estimation of the Markov random field. Alternatively, a similar method was applied to the related problem of classification [14], but only presented numerical results.

Incentivizing relaying has been well studied. A popular method is to use a payment scheme to reward users for relaying others' traffic [8]. In these schemes, a virtual currency is distributed among the users, and users charge others to relay for them. Another approach is reputation-based [9]. These schemes have either a centralized or distributed mechanism to track the reputation of each user, that is, how much relaying each user has done for others. A key part of these systems is ensuring their truthfulness. However, as D2D communications are typically between humans, utilizing social ties can allow for users to relay for other users they know.

Reducing cellular peak traffic caused by viral content is a rich field. Other works have considered knowing the resulting traffic pattern non-causally or using a Markov chain Monte Carlo approach [10] to predict the future traffic. Another method is to approximate the demand using the Bass diffusion model [11], which allows for a high-level proactive seeding design. However, online social networks drive viral content, and exploiting the known social ties can allow for a much better prediction of the future cellular traffic. Also, neither work has presented a method to identify exactly which user wants the content at what time, rather the number of users is the focus.

Chapter 2

MODELING SOCIAL TIE STRUCTURE VIA T-CHERRY JUNCTION TREES

2.1 Introduction

Over the past decade, online social networks have exploded in popularity. For instance, Facebook has grown to 1.11 billion users since its inception in 2004 [15]. People with smartphones are seemingly constantly connected to multiple online social networks, such as Twitter, Facebook, or Tumblr. Even the largest social networks continue to grow at a rapid pace. For example, in May of 2013, it was measured that approximately 135,000 new users signed up for Twitter every day [16].

The enormous scale of social networks highlights the need for a systematic quantification of social relationships of users that can be updated efficiently. Clearly, an exact characterization of the joint distribution of all possible social relationships in a social network is impossible in practice, and performing perfect inference would be computationally infeasible. To address these challenges, we propose a framework building on a recently developed tool in probabilistic graphical models to approximate this joint distribution: the t-cherry junction tree, first proposed in [17]. A t-cherry junction tree is a structure that has theoretical guarantees on accuracy of approximation, as well as allowing for efficient, exact inference once it is constructed. Other graphical models are often used, such as factor graphs [2], but these offer no guarantees on their approximation and may not convergence when performing inference [12]. *Notably, when using a junction tree, any dependence loops among random variables, which often occur in social relationships, can be easily handled by incorporating those random variables into a single cluster.* Further, the t-cherry junction tree is a data-driven approach, which does not require a specific model of user relationships. In a nutshell, this approach is tailored to offer a rigorous characterization of social ties, as opposed to approaches based on heuristics. We also demonstrate the flexibility of the t-cherry junction tree by developing methods to build a higher order t-cherry junction tree approximation without constructing it from scratch.

Given the massive scale, accurate mining and approximation of large social networks cannot be computed quickly on a single machine. The amount of data to be processed calls for parallel (cloud) computing. A vital strength of the junction tree structure is the ease in which building and utilizing this structure can be parallelized, which is another subject of this study.

To demonstrate the utility of this new approach for characterizing users' relationships, we investigate a specific application, namely recommending connections to new users of a social network. This is an important problem as online social networks remain committed to attracting and keeping new users. Once a new user has created an account, ensuring that this new user is actively connected to the online community is an important method to retain new users. A user with many connections in the social network receives more updates more frequently than one with only a handful of connections. Therefore, it is important to recommend potential connections in which a new user may be interested. As a new user begins to form online relationships, the probabilities of forming relationships to different users are likely to change due to the new information gained.

For this application, we seek to predict whether a user new to the online social network will form a relationship to others in the social network. For ease of reference, we call this problem the *new user recommendation problem*, which is related to the traditional "link prediction problem" [1]. In traditional link prediction, the set of users in the network remains fixed while links are added between them over a period of time.

Thus there are no new users to consider, and the timescale of interest is typically long as links may form slowly over time. Rather than having a fixed node set, the problem we study is to model the links that a new user, not a node previously in the network, will add to users currently in the network. In general, new users add connections in a social network much faster than long-time users. Thus, in the timescale under consideration, connections between users already in the network remain relatively static, whereas the new user will potentially make many new connections.

We investigate this problem from a purely topological viewpoint: we assume that the connections between users are the only data available. We do not incorporate attribute data of the users, as other works have [2] [3]. Once a new user begins to make connections to other users in the network, the probability that this user will connect with different users in the network must be updated so that the most likely social connections are recommended.

We summarize below the main contributions in modeling social relationships using a t-cherry junction tree.

- We develop a framework based on the t-cherry junction tree to approximate the underlying social relationships in a large-scale social network. Building on the greedy algorithm in [5], we construct a t-cherry junction tree by parallelizing most of the computations. Further, we present an exact method to test if each possible entry is valid by formalizing the characteristics of a t-cherry junction tree.

- We devise a scheme involving two algorithms to construct a higher order t-cherry junction tree in order to improve the approximation. This process is orders of magnitude faster than computing a higher-order tree from scratch, and indeed simulations indicate that the resulting higher order approximation reduces the

KL-divergence by 150% compared to the original one. The two main steps in this scheme are: I) the order update step, in which each cluster adds a random variable from one of its neighboring clusters, and hence changes a k-order *t-cherry junction tree* to a (k+1)-order *junction tree* (not necessarily a t-cherry junction tree); II) the t-cherry conversion process, which takes the resulting *junction tree* and refines each connection between clusters to reestablish the t-cherry property, while simultaneously greedily minimizing the Kullback-Leibler divergence between the approximation and the true distribution.

- This new framework is applied to the specific problem of recommending the most likely social connections to a new user by using a 100,000 user Twitter dataset. First, an existing tool, METIS [18], is used to partition the social graph into 1,560 disjoint subgraphs. For each of these subgraphs, a 4-order t-cherry junction tree is constructed, and then these individual trees are joined together using a slight variant of the t-cherry conversion process. The resulting KL-divergence between the t-cherry junction tree and the joint distribution is calculated explicitly, which is novel as the size of the joint distribution prevents a direct computation. Additionally, we leverage this structure to perform inference to update the probability of social relationships as new relationships form. Updating these probabilities is performed in a few minutes, despite involving 100,000 random variables.

The organization of the rest of the chapter is as follows. We begin with an introduction to the t-cherry junction tree in Section 2.2 and present a method for constructing it in Section 2.3. In Section 2.4, a process to gracefully improve the approximation given by a t-cherry junction tree is developed. The application of these tools to the new user recommendation problem is then presented in Section 2.5. The chapter concludes in Section 2.6.

2.2 T-Cherry Junction Trees

Denote the underlying joint distribution, containing N random variables, as $p(X_1, \ldots, X_N)$. When N is large, computing this joint distribution directly is generally too costly. Instead, a junction tree is created to approximate this distribution by using only "marginal" distributions corresponding to a small subset of random variables. Every junction tree has an underlying Markov random field that corresponds to the dependence structure inherent in the junction tree.

Definition 2.2.1. *A **junction tree** is a tree structure over a variable set $X_1, \ldots, X_N$ with the following properties:*

1. *Each node of the junction tree is a subset of random variables, denoted X_C, and is called a cluster. Associated with each cluster is the distribution $p(X_C)$. These clusters represent the maximal cliques of the associated Markov random field.*

2. *Every edge connecting two clusters in the junction tree contains a separator. A separator is a subset of random variables containing the intersection of the two clusters being linked: $X_S = X_{C_1} \cap X_{C_2}$. As with clusters, each separator has the distribution $p(X_S)$ associated with it.*

3. *If a random variable is contained in two different clusters, it is also contained in every cluster on the path between those two. This is called the running intersection property.*

4. *The union of all clusters is the entire set of random variables $X_1, \ldots, X_N$.*

Additionally, the **treewidth** of a junction tree is defined as the number of variables within the largest cluster. A slightly less restrictive concept than the junction tree is the **clique tree**. The difference is that a clique tree does not necessarily satisfy the running intersection property.

In order to characterize the quality of the junction tree approximation, we denote the entropy of a random variable X as $H(X)$ and the entropy of a random vector $\boldsymbol{X}$ as $H(\boldsymbol{X})$. Also, define the information content of a random vector $\boldsymbol{X} = [X_1, \ldots, X_k]$ as

$$I(\boldsymbol{X}) = \sum_{\boldsymbol{X}} p(\boldsymbol{X}) \log \left(\frac{p(\boldsymbol{X})}{p(X_1) \ldots p(X_k)} \right). \tag{2.1}$$

Using these definitions, the quality of the approximation is given by the following result [5].

Lemma 2.2.1. *The Kullback-Leibler divergence between a junction tree approximation and the actual distribution is:*

$$D_{KL}(p(\boldsymbol{X}) || p_{JT}(\boldsymbol{X})) = -H(\boldsymbol{X}) - \sum_{X_C \in \mathcal{C}} I(X_C) + \sum_{X_S \in \mathcal{S}} I(X_S) + \sum_{i=1}^{N} H(X_i). \tag{2.2}$$

Noting that the first and last terms of the expression do not depend on the junction tree, the closest approximation for a fixed treewidth is formed by constructing the set of clusters and separators to maximize the weight of the junction tree, defined as

$$w \triangleq \sum_{X_C \in \mathcal{C}} I(X_C) - \sum_{X_S \in \mathcal{S}} I(X_S). \tag{2.3}$$

However, finding the tree that maximizes the weight is an NP-hard problem [5]. Despite this challenge, there is a subclass of junction trees that is guaranteed to contain the maximum weight junction tree: t-cherry junction trees.

The t-cherry junction tree was originally derived from the t-cherry tree [19] and the simplex m-multitree [20]. We define the t-cherry junction tree in a slightly different, though equivalent, manner than it was originally presented in [17].

Definition 2.2.2. *The* ***k-order t-cherry junction tree*** *is a junction tree with the following properties:*

1. *Each cluster contains k random variables.*

2. *Every separator contains k-1 random variables.*

Thus two connected clusters, denoted C_1 and C_2, each contain a single random variable not contained in the other cluster. That is, $|C_1 \setminus C_2| = 1$, and $|C_2 \setminus C_1| = 1$.

We emphasize that the class of k-order t-cherry junction trees is important because the junction tree with the largest weight and hence the closest approximation to the actual distribution, is a member of this class [5].

Lemma 2.2.2. *In the class of all k-width junction trees, the k-order t-cherry junction tree having the greatest weight is the best approximation to the actual distribution.*

2.3 Data-Driven Construction of a t-Cherry Junction Tree

As the class of t-cherry junction trees is guaranteed to contain the best possible approximation among all k-order junction trees, we next investigate how to build a t-cherry junction tree. To this end, we develop an algorithm and show that the running time of this algorithm can be drastically reduced via parallel computation.

As mentioned above, constructing an optimal k-order t-cherry junction tree is an NP-hard problem. Based on [21], we design a greedy algorithm to construct a k-order t-cherry junction tree. It is worth noting that there are two key issues we resolve here. First, the process of checking if a cluster-separator pair is a valid addition is now explicitly formulated, and efficiently implemented by two checks. Our second contribution is to parallelize the process of listing all possible cluster-separator pairs, which allows the tree to be constructed significantly faster.

The basic idea of this algorithm is as follows. It begins by listing all possible clusters-separator pairs, and then orders them by weight. It selects the heaviest cluster-separator pair and adds it to the junction tree if the resulting clique tree is a valid t-cherry junction tree. If that cluster-separator pair is not a valid addition, it is discarded and the next heaviest pair is evaluated. This algorithm is motivated by the Chow-Liu algorithm for constructing a second order approximation [22], and indeed this algorithm is exactly the Chow-Liu algorithm when k is set to two. The output of this algorithm is a t-cherry junction tree, represented by a set of clusters $\mathcal{C}$ and set of separators $\mathcal{S}$, and a listing of the parent for each cluster. As clusters are added to the t-cherry junction tree, these sets are updated.

As presented in Algorithm 1, the construction has two phases: the first phase is **table construction**, and the second is **cluster addition**. It begins with the table construction phase. First, a table T listing all of the $k\binom{N}{k}$ possible cluster-separator pairs is constructed. That is, a list is made of all of the $\binom{N}{k}$ subsets of size k as clusters, and for each cluster, all of the k possible choices for the separator are listed. Denote each cluster-separator pair entry, with the cluster labeled as C' and the separator as S', in table T as follows: set $T(i,1) = C' \setminus S'$ and $T(i,2) = S'$. The single variable $C' \setminus S'$ is called the **dominating vertex** of the cluster [20]. For each of the $k\binom{N}{k}$ entries in table T, calculate the weight of the entry as $w = I(X_{C'}) - I(X_{S'})$ and set $T(i,3) = w$. Next, this table is sorted by heaviest weight, and the sorted table is labeled $T^\star$.

After the table $T^\star$ is constructed, the cluster addition phase takes place. Starting with an "empty" junction tree, determine whether the heaviest remaining entry can be added to the junction tree. Each cluster, aside from the first cluster entered, will have a parent associated with it. To facilitate checking if the cluster-separator addition is valid, maintain a binary vector of all nodes currently represented in the junction tree. This vector is denoted V, where $V(v) = 1$ represents that variable v is contained in at least one cluster in the junction tree.

Next, we examine the condition under which a cluster-separator pair can be added and maintain a valid t-cherry junction tree. The variables in S' must be contained within another cluster to have a valid parent. If the dominating vertex of C' is already in a cluster of the network, then this cluster-separator pair is not a valid addition because there are only two possible manners in which this occurs, neither of which results in a valid t-cherry junction tree. The first possibility is that the dominating vertex is also contained within the parent cluster. This results in the two clusters being identical, which is not valid as the separator would contain k variables. The other possibility is that the dominating vertex is contained within a cluster in the tree that is not the parent cluster. As the dominating vertex is not in S', the running intersection property would be invalidated.

Though this approach is heuristic and cannot guarantee that the t-cherry junction tree created has the global maximum weight, it has the benefit of being largely parallelizable to decrease the overall running time. Other methods for constructing a junction tree have been proposed, but these methods operate by selecting links in the underlying Markov random field individually in a step-by-step manner [23] [24]. However, these methods cannot be parallelized and are applicable to only small datasets.

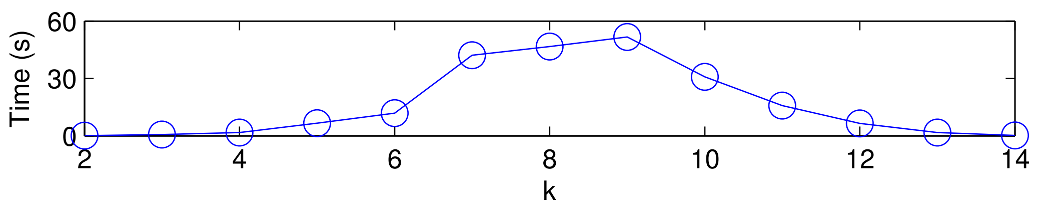
It can be shown that the overall time to construct a junction tree is dominated by the table construction phase on average, and therefore decreasing the time needed to construct the table will greatly decrease the time needed to build a t-cherry junction tree. To get a more concrete sense, we run the greedy algorithm using a Twitter data set [25], using 15 binary random variables and vary the width of the tree generated from 2 to 14. From this Twitter data set, we select the 500 nodes with the most

Algorithm 1 Greedy Algorithm for Constructing a k-Order t-Cherry Junction Tree

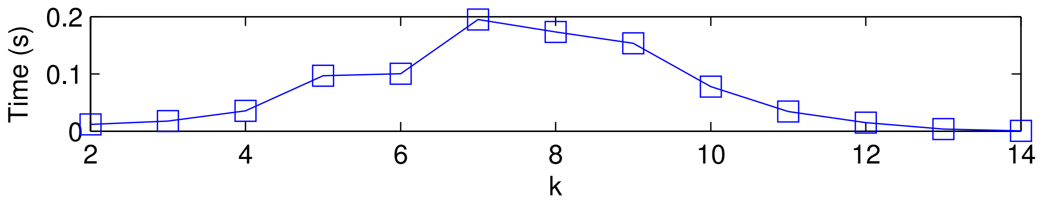
```
$\mathcal{C} = T^\star(1,1) \cup T^\star(1,2)$
$\mathcal{S} = \varnothing$
$V\left(T^\star(1,1) \cup T^\star(1,2)\right) = 1$
$i = 2$
while $\exists v$ such that $V(v) = 0$ do
    $d' = T^\star(i,1)$
    $S' = T^\star(i,2)$
    $C' = d' \cup S'$
    if $V(d') = 0$ then
        for $j = 1 : |\mathcal{C}|$ do
            $C = \mathcal{C}(j)$
            if $S' \subset C$ then
                parent$(C') = C$
                $\mathcal{C} = \mathcal{C} \cup C'$
                $\mathcal{S} = \mathcal{S} \cup S'$
                $V(d') = 1$
                break loop
            end if
        end for
    end if
    $i = i + 1$
end while
```

connections and randomly select 15 nodes for each iteration. The results for each treewidth were averaged over 200 runs. The average times needed to generate the table for different treewidths and the average time to construct the junction tree are shown in Figure 2.1. Clearly, the time to generate the table is several orders of magnitude larger than the time needed to construct the junction tree for all treewidths.

As the performance bottleneck of this approach is the table construction phase, we use parallel computing to decrease the runtime. This parallel computation can be exported via cloud computing [26] in order to drastically reduce the time needed to construct a t-cherry junction tree. Each entry of the table can be computed separately from every other entry and then these individual entries can be sorted by weight to form the complete table.



(a) Table construction



(b) Cluster addition

Figure 2.1: Time of each phase of building a t-cherry junction tree

2.4 A Closer Approximation: from k-Order to (k+1)-Order

While low order t-cherry junction trees can be constructed quickly, constructing high order ones from scratch would be formidable. In order to improve the approximation generated by a low order t-cherry junction tree, we propose an iterative method to update a k-order t-cherry junction tree to (k+1)-order gracefully, which refers to using an incremental process based on the previously constructed k-order tree. There are two steps to this process. The first step is the **order update** process, by which each cluster is expanded from k variables to k+1 variables. This step maintains the junction tree structure, but cannot guarantee that the t-cherry property is retained. Reestablishing the t-cherry property is the second step, namely via **t-cherry conversion**.

2.4.1 The Order Update Process

Clearly, the first step of updating from a given k-order t-cherry junction tree to a (k+1)-order junction tree is to add a “good” variable to each cluster. We develop an algorithm to add a variable to each cluster by including a variable contained in a neighboring cluster. The variable added to a cluster must be from one of its neighboring clusters in order to satisfy the running intersection property, or else the result would be a clique tree, not a junction tree.

The algorithm we design to perform this process is called the **Order Update Algorithm**, and it operates by implementing a series of **update steps**, each of which is the process of adding a variable to a cluster, until each cluster in the junction tree is of size k+1. Each update step consists of adding an **additional variable** to an **eligible cluster**, a cluster which has not yet added a variable. Each cluster can only add a single variable as the treewidth of the updated junction tree is k+1. Once an eligible cluster has added a variable, it becomes an **ineligible cluster**.

To determine which variable is the “best” variable for a cluster to add, each potential update step has a weight assigned to it. This weight represents the increase in the weight of the junction tree after the update step takes place. To calculate this weight, note that the update step process is equivalent to adding a new edge in the underlying Markov random field. This insight allows the weight to be calculated efficiently using a result from [23]. To formulate this result, and for ease of exposition, for the rest of this section we denote the **active cluster**, the cluster adding the variable, as C_A and the **donating cluster**, the neighboring cluster that contains the additional variable, C_D. The separator of these clusters is S and the additional variable is v. After the update step is performed, we denote the resulting cluster as $C'_A \triangleq C_A \cup v$. The edge to be added in the Markov random field is between the additional variable v and the single node $d \triangleq C_A \setminus C_D$. Utilizing Corollary 4.1 in [23], the weight increase of the junction tree due to an update step is:

$$\Delta W(d, v) = H(S \cup d) + H(S \cup v) - H(S \cup d \cup v) - H(S). \tag{2.4}$$

There are three steps that need to be performed for each update step. The first is to simply include the additional variable in the active cluster, and mark the active cluster as ineligible. Now that the active cluster contains a new variable, each neighbor of the active cluster has a new potential update step as the current additional variable v could be added to each neighbor, which was not possible before. Note that the donating cluster does not have a new potential update step because it already contains the current additional variable. So the second step is to find all new potential update steps for the neighbors of the active cluster.

The third step is more involved than the previous two. As two connected clusters differ by only a single variable, some update steps will result in the donating cluster becoming a subset of the active cluster. When this occurs, the donating cluster must be removed from the junction tree, and its neighbors must be connected to the active cluster instead. After these connections are migrated to the active cluster, some potential update steps of each neighbor of the donating cluster must be changed to reflect that any potential update step that used C_D as the donating cluster must now use C'_A as the donating cluster. Also, each of these new neighbors of C'_A has a new potential update step in which the additional variable would be d.

We now present the Order Update Algorithm as Algorithm 2. At the beginning, a list of all potential update steps is constructed, denoted T, where each entry is of the form $\{w, v, C_A, C_D\}$, which represents cluster C_A adding variable v from cluster C_D with the resulting weight increase of the junction tree being w. This list is sorted by weight. Next, update steps are executed in a greedy manner by performing the update step with the greatest w, but only if the active cluster is eligible. If not, this entry is simply discarded. After each update step, the three necessary steps previously outlined are carried out. The algorithm continues until all clusters are marked ineligible. To track the eligibility of clusters, there is a binary vector over the set of clusters denoted E, where $E(i) = 1$ represents that cluster C_i is eligible. The entire set of clusters in the junction tree is labeled $\mathcal{C}$ and the set of all neighbors of a cluster C_i is denoted $\mathcal{N}(C_i)$.

An example demonstrating a single update step to move from a 4-order to a 5-order junction tree is presented in Figure 2.2 for a small junction tree. After one update step, the cluster $\{X_1, X_2, X_3, X_4\} \triangleq C_1$ adds variable X_6 from its neighboring cluster $\{X_1, X_3, X_4, X_6\} \triangleq C_2$ to become cluster $\{X_1, X_2, X_3, X_4, X_6\} \triangleq C_1'$. Because $C_2 \subset C_1'$, C_2 is removed and all connections from C_2 to any clusters aside from C_1' are then connected to C_1'. Note that the separators from all neighbors of C_1 and C_2 remain unchanged after the update process. There are three updates that are made to the list of possible update steps. The first is that cluster $\{X_1, X_2, X_3, X_5\}$ can now add variable X_6 from cluster C_1', so this entry is added to the list. The second addition is that cluster $\{X_1, X_4, X_6, X_7\}$ can now add variable X_2 from cluster C_1'. The third is that cluster $\{X_1, X_4, X_6, X_7\}$ can still add variable X_5, but the entry in T needs to be changed so that the cluster from which it adds this variable is set to C_1', not C_2 as it was previously set. In total, there are two additions to the list of possibilities, and one change.

It is important to note that after each update step, the resulting clique tree is still a junction tree.

Theorem 2.4.1. *Each update step preserves the junction tree structure, i.e., it does not reduce a junction tree to merely a clique tree.*

The proof of this theorem is found in Section 2.7.

Algorithm 2 Order Update Algorithm

```
1:  $E(C_i) = 1 \; \forall \; C_i \in \mathcal{C}$
2:  while $\exists i$ such that $E(i) = 1$ do
3:    $C_A = T(1,3)$
4:    if $E(C_A) = 1$ then
5:      $v = T(1,2)$
6:      $C_D = T(1,4)$
7:      $d = C_A \setminus C_D$
8:      $C_A = C_A \cup v$
9:      $E(C_A) = 0$
10:     for all $C_i \in (\mathcal{N}(C_A) \setminus C_D)$ and $E(C_i) = 1$ do
11:       $v' = C_i \setminus C_A$
12:       $w = \Delta W(v', v)$
13:       Add entry $\{w, v', C_i, C_A\}$ to table $T$
14:     end for
15:     if $C_D \subset C_A$ then
16:       for all $C_i \in (\mathcal{N}(C_D) \setminus C_A)$ do
17:         Connect $C_i$ to $C_A$
18:         $v' = C_i \setminus C_A$
19:         $w = \Delta W(d, v')$
20:         Add entry $\{w, d, C_i, C_A\}$ to table $T$
21:       end for
22:       for all $T_i \in T$ do
23:         if $T_i(4) = C_D$ then
24:           $T_i(4) = C_A$
25:         end if
26:       end for
27:       Remove cluster $C_D$ from the junction tree
28:     end if
29:   end if
30:   Delete the first entry in $T$
31:   Sort $T$ by $w$
32: end while
```

2.4.2 The t-Cherry Conversion Process

Though the previous algorithm returns a (k+1)-order junction tree, it does not necessarily return a (k+1)-order t-cherry junction tree. The reason for this is that the k-order t-cherry junction tree has separators that contain k-1 variables, so a (k+1)-order junction tree must have separators that contain k variables. However, the separators in the junction tree returned from the algorithm may have separators of size k-1. This occurs when two connected clusters, C_1 and C_2, have added variables from other clusters, i.e., C_1 added a variable from a cluster other than C_2, and vice-versa. This implies that the separator $S = C_1 \cap C_2$ is unchanged from its original k-1 nodes, as the intersection of the two clusters remains the same.

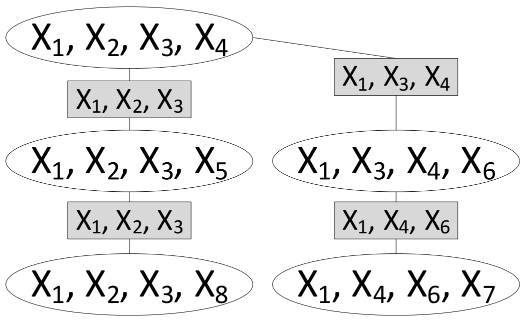
We next present an algorithm that converts the (k+1)-order junction tree returned from Algorithm 2 into a (k+1)-order t-cherry junction tree. We note that this algorithm can be extended in a straightforward manner to convert any (k+1)-order junction tree, not only those returned by Algorithm 2, into a (k+1)-order t-cherry junction tree. The reason for the difference is that a general (k+1)-order junction tree may have clusters of any size between 1 and k, as the treewidth is the size of only the largest cluster in the entire junction tree, whereas in the (k+1)-order junction tree returned by Algorithm 2, all clusters are of size $k + 1$.

An earlier algorithm [5], developed to convert a junction tree to a t-cherry junction tree, requires that a root node be selected, and the resulting tree is heavily dependent upon the choice of root node. Our approach does not require a root node to be selected, and it greedily maximizes the weight of the junction tree as it operates. Also, the underlying process, maintaining separators of size k, is clearly articulated in our algorithm.

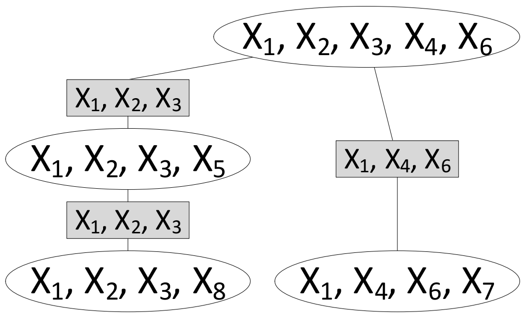
The t-cherry conversion process is completed when each separator has been transformed to include k variables. Thus we design an algorithm to ensure that each separator contains k variables, and we call this the **t-Cherry Bud Conversion Algorithm**, presented as Algorithm 3. A **bud** of the t-cherry junction tree is a cluster-separator-cluster triplet, and thus the number of buds is the number of separators in the junction tree. For each separator in the junction tree, the bud can be replaced by a set of clusters and separators that have the t-cherry property and still remain a junction tree. This process transforms each bud in turn, until the entire junction tree has been converted to a (k+1)-order t-cherry junction tree. In other words, the t-cherry conversion process operates by transforming all buds into t-cherry compliant buds by transforming each using Algorithm 3.

Algorithm 3 operates over two clusters and their separator. Denote one cluster as C_1, the other C_2, and S as their separator. The underlying concept is to "pull forward" variables from C_1 to create more clusters in order to ensure that adjacent clusters are separated by k variables (as the treewidth is k+1) and that the running intersection property is maintained. In total, the two original clusters are replaced by $|C_2| - |S| + 1$ clusters, as each variable contained in $C_2 \setminus S$ must be a dominating vertex of its own cluster. Denote the set of dominating vertices not yet assigned to clusters as $\mathcal{D}$, and the set of dominating vertices already placed in clusters as $\mathcal{E}$. At the beginning of the algorithm, $\mathcal{D} = C_2 \setminus S$ and $\mathcal{E} = \varnothing$. A key component of this algorithm is the set of "free" variables; that is, the set of variables that can be pulled forward from the previous cluster to a new cluster to ensure the separator is of size k. We denote this set of free variable as $\mathcal{F}$. The first iteration is slightly different from the next $|C_2| - |S|$ iterations so it is outside of the main loop. The sets of the new clusters and separators returned are denoted $\mathcal{C}'$ and $\mathcal{S}'$ respectively. These two sets replace C_1, S, and C_2 in the junction tree. For each iteration, the choice of F and d are made to maximize the weight of the resulting cluster-separator pair, as seen in lines 4 and 11.

An example of this process is shown in Figure 2.3. Figure 2.3(a) shows a 5-order junction tree, after the order update process that began in Figure 2.2 has completed all of its steps. Notice that the right branch already satisfies the t-cherry property,

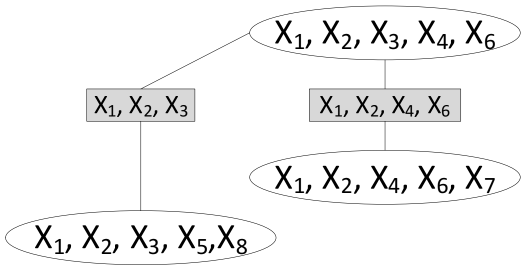


(a) Original

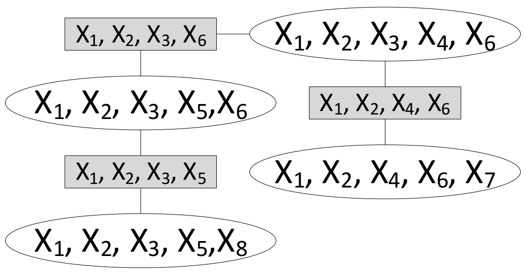


(b) After update step

Figure 2.2: An example of one update step



(a) After order update process



(b) After t-cherry conversion

Figure 2.3: An example of t-cherry conversion

so it does not need to be converted. However, the left branch, the cluster-separator-cluster connection $\{X_1, X_2, X_3, X_4, X_6\}$ $\{X_1, X_2, X_3\}$ $\{X_1, X_2, X_3, X_5, X_8\}$, does not satisfy the t-cherry property as the separator has only 3 variables, not 4. The result after running Algorithm 3, shown in Figure 2.3(b), has an additional cluster inserted between the two original clusters to make a t-cherry junction tree.

In order to convert all buds into t-cherry buds, Algorithm 3 is run twice on each bud. For the second application of this algorithm, the choice of C_1 and C_2 is reversed. After both cluster and separator sets are calculated, the choice that results in the heaviest weight is used to replace the bud in the junction tree.

Algorithm 3 t-Cherry Bud Conversion Algorithm

```
Require: $C_1, C_2, S = C_1 \cap C_2$
 1: $\mathcal{D} = C_2 \setminus C_1$
 2: $\mathcal{C}'_1 = C_1$
 3: $\mathcal{F} = \{F : F \subset C_1, |F| = k - |S|\}$
 4: $F^\star, d^\star = \underset{F \in \mathcal{F}, d \in \mathcal{D}}{\arg\max}\ I(d \cup S \cup F) - I(S \cup F)$
 5: $\mathcal{C}'_2 = d^\star \cup S \cup F^\star$
 6: $\mathcal{S}'_1 = S \cup F^\star$
 7: $\mathcal{D} = \mathcal{D} \setminus d^\star$
 8: $\mathcal{E} = d^\star$
 9: for $i = 2 : (|C_2| - |S|)$ do
10:     $\mathcal{F} = \{F : F \subset \left(\mathcal{S}'_{i-1} \setminus (\mathcal{E} \cup S)\right), |F| = k - |\mathcal{E}| - |S|\}$
11:     $F^\star, d^\star = \underset{F \in \mathcal{F}, d \in \mathcal{D}}{\arg\max}\ I(d \cup \mathcal{E} \cup S \cup F) - I(\mathcal{E} \cup S \cup F)$
12:     $\mathcal{C}'_{i+1} = d^\star \cup \mathcal{E} \cup F^\star \cup S$
13:     $\mathcal{S}'_i = \mathcal{E} \cup F^\star \cup S$
14:     $\mathcal{D} = \mathcal{D} \setminus d^\star$
15:     $\mathcal{E} = \mathcal{E} \cup d^\star$
16: end for
17: return $\mathcal{C}', \mathcal{S}'$
```

As with Algorithm 2, it is vital to show that this approach returns not only a junction tree structure, not merely a clique tree, but also specifically a t-cherry junction tree.

Theorem 2.4.2. *The clique tree, constructed via performing Algorithm 3 twice on each bud, is a t-cherry junction tree.*

The proof of this result can be found in Section 2.8.

To demonstrate the improvement from updating a k-order t-cherry junction tree to a (k+1)-order t-cherry junction tree, we apply this approach to each of the 1,560 individual t-cherry junction trees constructed in Section 2.5. The average increase in the weight of the junction tree after the order update step is 19%, and the average weight increased another 110% on top of the order update weight after performing the t-cherry conversion process. Thus after both steps, on average the weight of the t-cherry junction tree increased 150%. Both the order update and t-cherry conversion processes combined required five seconds on average to run for each individual tree, a significantly shorter amount of time compared to building a 5-order t-cherry junction tree from scratch, which takes roughly one and a half hours. This approach is a very fast method to significantly improve the approximation of a t-cherry junction tree.

2.5 An Example Application: New User Recommendation in a Twitter Network

Next we apply the tools developed in the previous sections to the new user recommendation problem. A t-cherry junction tree is constructed to approximate the joint distribution of social connections in the social network, and then this junction tree is used to perform inference to update the probabilities of social links forming. Our experimental setup is to approximate the probability of following users in a set of 100,000 Twitter users using the dataset [25]. We first preprocess the data and extract the 100,000 most connected users. The random variables in this application are whether a new user will follow a certain user. The joint distribution $p(X_1, \ldots, X_N)$ is a collection of $N = 100,000$ binary random variables, where $X_i = 1$ represents the outcome that a new user to Twitter will follow user i. To construct a t-cherry junction tree, the "marginal" distributions containing k and $k-1$ variables must be computed. Rather than attempting to model human social behavior directly, we use a data driven approach instead. Each of these distributions is computed using the empirical data by counting the number of the 100,000 users that follow each user.

As constructing a t-cherry junction tree requires constructing a table with $k\binom{N}{k}$ entries, a small treewidth, $k = 4$, is chosen, as is typical for constructing junction trees. Due to the fact that directly constructing a 100,000 variable t-cherry junction tree is computationally infeasible, the social graph is partitioned using METIS [18], specifically the implementation [27], into 1,560 disjoint subgraphs, with each subgraph containing 64 or 65 users. We call the t-cherry junction tree constructed for each subgraph an **individual tree**. Each individual t-cherry junction tree can be computed in a reasonable amount of time, roughly five minutes, using Algorithm 1. It is necessary to partition the graph into *disjoint* subgraphs in order to ensure that when the individual t-cherry junction trees are joined together, the running intersection property is not violated. The METIS software partitions by attempting to minimize the number of edges cut to form the partitions. Retaining the maximum number of edges in the partitions is ideal as attempting to keep densely connected social communities together is important. In order to build these individual trees, 7 computers were used operating in parallel.

Next, we examine the process of connecting the individual t-cherry junction trees together in order to form one connected t-cherry junction tree. Two individual t-cherry junction trees can be joined together by connecting two clusters, one from each tree, using Algorithm 3 to ensure that the resulting combination is still a t-cherry junction tree. It is important that only two clusters are joined together; otherwise loops will be present, and thus it will not be a junction tree. There are two sub-problems in this process. The first is that each individual tree can be joined with any other tree, and the second is that after choosing which two individual trees to join together, each cluster within each tree is a potential choice to use in connecting these two trees.

To attempt to maintain the original social structure of the full social graph, the number of edges cut in the partitioning process is the metric used to decide which individual trees to be joined. The process begins by computing the number of edges cut between all partitions, and then combining the two individual trees that have the maximum number of edges cut between them. This process continues in a greedy manner by selecting the individual tree that has the most edges cut to any of the already joined individual trees. This continues until all of the individual t-cherry junction trees have been joined together.

Having selected which individual t-cherry junction trees to join, the other subproblem is to decide which two clusters should be connected to actually join the junction trees together. To join these individual junction trees together, the social structure of the original graph was considered. For each two individual trees being joined together, the number of edges cut between the users in each cluster in both trees was counted. Then the two clusters which had the most edges cut between them were selected to be connected in order to join the two individual trees. After selecting these two clusters, they are linked together using Algorithm 3, which ensures that the t-cherry property is preserved. After all individual trees were joined together in this manner, the result was a single t-cherry junction tree containing 100,000 variables. This t-cherry junction tree captures the joint probability distribution of a new user following any of the existing 100,000 users.

The entire tree building process, from initial partitioning through the combination of the individual trees took slightly over 19 hours, the vast majority of which was spent constructing the individual junction trees. To calculate the KL-divergence between this approximation and the joint distribution, we calculate the overall weight of this combined junction tree as 926. The estimated joint entropy is 11.5 and the sum of the individual entropies is 12,596. Using Equation (2.2), the KL-divergence of this approximation is 13,511. While this represents a large difference between the approximation and the joint distribution, this KL-divergence is still calculable, which is novel, and this approximation uses only 99,997 distributions containing 4 variables, and 99,996 distributions containing 3 variables, resulting in a total of 2,399,920 data entries to store these distributions, to model a distribution containing 100,000 variables. Compared to storing the full distribution, which requires $2^{100,000}$ entries, this is a very small number of entries.

With the complete t-cherry junction tree constructed, the final piece of the solution to the new user recommendation problem can be developed. This last piece is the inference process. Once a new user has begun to select other users to follow, the values of the random variables representing these relationships are known, and the probability of the unknown relationships can then be updated.

Computing this new distribution, performing inference, for a junction tree can be evaluated in a computationally efficient manner by using message passing to update all clusters to their new distributions [28]. This process is quick and there are other works to parallelize this process [29], and the software suite GraphLab can be used as an off-the-shelf implementation [30]. For the full junction tree containing all 100,000 users, inference takes less than two minutes. We test our approach by calculating the area under the ROC curve (AUC) by testing 1,000 random users contained in the Twitter dataset, but not in the 100,000 users used to build the junction tree. The resulting AUC is 0.5519, which is impressive considering the massive scale of this social network and the low order of the t-cherry junction tree.

2.6 Conclusion

In this chapter, we developed a framework based on the t-cherry junction tree to characterize users' relationships in online social networks. To this end, we devised an algorithm to construct a k-order t-cherry junction tree where most of the computations are parallelized. In order to improve the approximation further, we proposed a scheme consisting of the order update and t-cherry conversion steps to construct a higher order t-cherry junction tree. The proposed scheme is significantly faster than building a higher order t-cherry junction tree from scratch and greatly decreases the

KL-divergence between the approximation and the joint distribution compared to the original one. This new framework was applied to the new user recommendation problem by creating a probabilistic model of 100,000 user relationships in a Twitter dataset by building 1,560 individual t-cherry junction trees and connecting them together.

To the best of our knowledge, this work is the first using a t-cherry junction tree approach to study social networks. In general, the junction tree has many strengths that allow for a compact and rigorous characterization of the underlying structure of social networks.

2.7 Proof of Theorem 2.4.1

It suffices to check all conditions in Definition 2.2.1. Denote the active cluster as C_1 and the neighboring cluster containing the additional variable v as C_2, with the separator S. Additionally, denote the new cluster of size k+1 as C_1', and the set of all clusters connected to C_1, excluding C_2, as $\mathcal{N}_1$, where these neighboring clusters are labeled $N_{1_i} \in \mathcal{N}_1$, and likewise for nodes connected to C_2, excluding C_1, as $N_{2_i} \in \mathcal{N}_2$.

The first condition of the definition is trivially satisfied. To check the second condition, we show that the separator S is correctly constructed, and all other separators remain unchanged. If $C_2 \not\subset C_1'$, then $S' \triangleq C_1' \cap C_2 = S \cup v$. Next, we demonstrate that each N_{1_i} is connected to C_1' with the same separator it originally had, which we denote $S_{1_i} = C_1 \cap N_{1_i}$. First, no variables need to be removed from S_{1_i} as all variables in C_1 are in C_1'. Secondly, no variables need to be added to S_{1_i} due to the fact that the running intersection property of the original tree guarantees that $v \notin N_{1_i}$. Thus S_{1_i} remains unchanged for every i. If C_2 is removed from the junction tree, we show that each $S_{2_i} \triangleq C_2 \cap N_{2_i}$ remains the same despite connecting to C_1' instead of C_2. As $C_2 \subset C_1'$, no variables need to be removed from this separator. No variables need to be added as the running intersection property of the original tree ensures that C_1' and S_{2_i} do not share any variables that were not in C_2.

The third condition, the running intersection property, clearly still holds if $C_2 \not\subset C_1'$. If $C_2 \subset C_1'$, each N_{2_i} now connected to C_1' still satisfies the running intersection property because every variable in C_2 is contained in C_1'. So any path between clusters with a shared variable that originally traversed C_2 now traverses C_1' with that same variable. And clearly the fourth condition is satisfied as no variables were removed from the junction tree.

2.8 Proof of Theorem 2.4.2

We begin with proving that this approach preserves the junction tree structure by checking the conditions of Definition 2.2.1. Denote the original clusters C_1 and C_2 and their separator as S. The resulting replacement clusters are labeled $C_{r_1}, \dots, C_{r_{|C_2|-|S|+1}}$.

The first and second conditions of the definition are clearly satisfied. In order to prove the running intersection property, note that $C_{r_1} = C_1$, and likewise $C_{r_{|C_2|-|S|+1}} = C_2$. Any path originally terminating on C_1 or C_2 still has the running intersection property. Therefore, once it is proved that $C_{r_1}, \dots, C_{r_{|C_2|-|S|+1}}$ satisfy the running intersection property internally, the overall running intersection property holds. This property is violated if and only if C_{r_i} contains a variable present in $C_{r_j} \setminus C_{r_{i-1}}$ $\forall j < i-1$. The dominating vertex of C_{r_i}, $d_i^\star$ is not contained in any previous vertex due to the fact that is was selected from $\mathcal{E}$. So only the remaining k nodes of C_{r_i} could possibly be contained within C_{r_j}. The original separator S is contained within every cluster, thus only the remaining $k - |S|$ variables of C_{r_i} could possibly be contained within $C_{r_j} \setminus C_{r_{i-1}}$. The dominating vertex of $C_{r_{i-1}}$, $d_{i-1}^\star$, is contained within $\mathcal{E}$ when C_{r_i} is constructed and is not contained in any cluster C_{r_j} $j < i-1$ by the construction of $\mathcal{E}$. Additionally, any previous dominating vertex is contained within $\mathcal{E}$ for the next iteration and is thus is all future clusters. Therefore only $k - |S| - |E|$ variables remain as candidates for variables contained in $(C_{r_i} \cup C_{r_j}) \setminus C_{r_{i-1}}$. These remaining variables are precisely the variables chosen from the "free variable" set $\mathcal{F}$. However, by the definition of $\mathcal{F}$, all possible $F \in \mathcal{F}$ are a subset of $C_{r_{i-1}}$. Thus there are no variables that could be present in $C_i \setminus C_{i_1}$ that are present in C_{r_j} $\forall j < i-1$. Therefore the running intersection property is satisfied. The fourth condition of the definition holds as no variables are removed.

As each call of Algorithm 3 preserves the junction tree structure, the entire process also preserves the junction tree structure.

To prove that a t-cherry junction tree structure is returned, note that if each call of Algorithm 3 returns a t-cherry junction tree, then the resulting junction tree from transforming each bud is a t-cherry junction tree. A (k+1)-order junction tree is a (k+1)-order t-cherry junction tree if and only if each separator $S = C_1 \cap C_2$ contains k nodes, i.e., C_1 and C_2 each contain only a single node that is not contained within the other cluster. This is clear to see as:

$$\begin{aligned}
C_{r_{i+1}} \setminus C_{r_i} &= (d^\star_{i+1} \cup \mathcal{E}_{i+1} \cup F^\star_{i+1} \cup S) \setminus (d^\star_i \cup \mathcal{E}_i \cup F^\star_i \cup S) \\
&= d^\star_{i+1} \cup (\mathcal{E}_{i+1} \setminus (d^\star_i \cup \mathcal{E}_i)) \cup (F^\star_{i+1} \setminus F^\star_i) \qquad (2.5) \\
&= d^\star_{i+1}. \qquad (2.6)
\end{aligned}$$

Similarly,

$$\begin{aligned}
C_{r_i} \setminus C_{r_{i+1}} &= (d^\star_i \cup \mathcal{E}_i \cup F^\star_i \cup S) \setminus (d^\star_{i+1} \cup \mathcal{E}_{i+1} \cup F^\star_{i+1} \cup S) \\
&= ((d^\star_i \cup \mathcal{E}_i) \setminus \mathcal{E}_{i+1}) \cup (F^\star_i \setminus F^\star_{i+1}) \qquad (2.7) \\
&= F^\star_i \setminus F^\star_{i+1}. \qquad (2.8)
\end{aligned}$$

Note that $|F_i^\star \setminus F_{i+1}^\star| = 1$ as

$$|F_i^\star \setminus F_{i+1}^\star| = k - |\mathcal{E}_i| - |S| - (k - |\mathcal{E}_{i+1}| - |S|) \tag{2.9}$$

$$= |\mathcal{E}_{i+1}| - |\mathcal{E}_i| = 1. \tag{2.10}$$

Therefore each pair of connected clusters differs by only a single node in each.

Chapter 3

A DATA DRIVEN APPROACH FOR MODELING DEPENDENCE STRUCTURE

3.1 Introduction

In traditional distributed detection problems, a collection of $M > 2$ distributed agents collect data and formulate decisions regarding a binary hypothesis test between common hypotheses H_0 and H_1. The decisions of the individual agents, represented as binary variates $\mathbf{U} \triangleq \{U_1, \ldots, U_M\}$, are aggregated at a fusion center, which forms a global decision regarding the two hypotheses on the basis of this data.

A prevalent assumption in addressing this problem is that the random variables U_i are conditionally independent under each hypothesis [31, 32]. This assumption is justified in many settings, and it yields a number of appealing outcomes; e.g., if the binary data are conditionally independent and identically distributed, the optimal decision rule at the fusion center is a simple voting scheme. And it is generally a weighted vote even when the data values are not conditionally identically distributed provided the probabilities of detection and false alarm for the local detectors are known [33].

Despite providing highly tractable fusion rules, the performance of the global detector that assumes independence may be poor if the sensors' measurements are correlated. In this scenario, exploiting the full joint conditional distributions of $\mathbf{U}$ at the fusion center yields optimal detection performance but introduces several drawbacks. One key impediment to working with full joint conditional densities of the binary data is that they are burdensome to represent and store. For M independent and identically distributed binary random variables and two hypotheses, the joint conditional densities are summarized by two parameters, which increases to $2M$ parameters if the assumption of identically distributed is removed. In the general case, 2^{M+1} parameters are required to catalogue a pair of joint conditional densities; with 50 sensors, this would be larger than 10^{15}. Another complication is the need to estimate the densities from data samples.

These two issues, namely the number of model parameters to store and the amount of training data necessary to accurately estimate them, point directly to the relative strengths and weaknesses of the two approaches. The first approach, considering all U_i as conditionally independent, involves storing only $2M$ parameters that can be accurately estimated using a small amount of training data. However, this may represent a very coarse approximation to the actual joint distribution of $\mathbf{U}$. The second approach is to consider the full joint distribution of the U_i, which is more accurate provided there is ample training data available to estimate the 2^{M+1} parameters, as well as storage space for these parameters.

In light of the limitations of these two approaches, we present a method that enables a controllable tradeoff between these two approaches. By constructing a t-cherry junction tree [5] for each hypothesis, approximations to the distribution of the U_i can be constructed of any order, rather than only the conditionally independent (first order) case, or the case with the full joint distribution (M^{th}-order). This approach offers flexibility and sheds light on the impact of the choice of the order on the approximation of the full joint distribution estimated from training data.

The outline of this chapter is as follows. The impact of using estimated distributions is quantified in Section 3.2. Simulations are used to demonstrate the performance of this approach in Section 3.3, and the paper concludes in Section 3.4.

3.2 Data-Driven Distribution Approximation

While the divergence between a t-cherry junction tree approximation and the actual distribution is understood when all distributions are known perfectly, we quantify the divergence between the full and junction tree distributions when the junction tree is constructed using distributions estimated from training data. The true distribution $P(\mathbf{U})$ remains the same, but now the t-cherry junction tree is constructed by first estimating all the necessary marginal distributions. Thus cluster and separator distributions are represented as $\hat{p}(\mathbf{U}_C)$ and $\hat{p}(\mathbf{U}_S)$, resulting in a joint distribution over all M random variables denoted $\hat{P}_{JT}(\mathbf{U})$. In this section, we calculate the scaling behavior of the increased KL divergence resulting from estimated distributions as a function of both the order of the t-cherry junction tree and the number of data samples.

To begin, we calculate the impact of using estimated distributions on the KL divergence between the true full distribution $P(\mathbf{U})$ and the junction tree approximation using estimated distributions $\hat{P}_{JT}(\mathbf{U})$.

Theorem 3.2.1. *The KL divergence between the true distribution $P(\mathbf{U})$ and the junction tree constructed using distributions estimated from training data $\hat{P}_{JT}(\mathbf{U})$ is*

$$\begin{aligned} D_{KL}(P(\mathbf{U})||\hat{P}_{JT}(\mathbf{U})) &= -H(\mathbf{U}) + \sum_{i=1}^{M} H(U_i) - \sum_{C\in\mathcal{C}} I(\mathbf{U}_C) + \sum_{S\in\mathcal{S}} I(\mathbf{U}_S) \\ &+ \sum_{C\in\mathcal{C}} D_{KL}(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)) - \sum_{S\in\mathcal{S}} D_{KL}(p(\mathbf{U}_S)||\hat{p}(\mathbf{U}_S)). \end{aligned} \tag{3.1}$$

Proof. We start with considering the definition of KL divergence between two absolutely continuous discrete distributions, which can be thought of as vectors of probabilities of length $|P(\mathbf{U})|$. Note that each cluster consists of k random variables and is written $\mathbf{U}_C = \{U_{C_1}, \ldots, U_{C_k}\}$ and likewise for separators. The KL divergence is then

$$D_{KL}(P(\mathbf{U})||\hat{P}_{JT}(\mathbf{U})) = \sum_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{P(\mathbf{U})}{\hat{P}_{JT}(\mathbf{U})} \right). \tag{3.2}$$

This divergence can be related to the divergence between the true distribution and the distribution from the junction tree created with the true, not estimated, marginal distributions, denoted $P_{JT}(\mathbf{U})$. This can be done by multiplying and dividing to yield

$$D_{KL}(P(\mathbf{U})||\hat{P}_{JT}(\mathbf{U})) = D_{KL}(P(\mathbf{U})||P_{JT}(\mathbf{U})) + \sum_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{P_{JT}(\mathbf{U})}{\hat{P}_{JT}(\mathbf{U})} \right). \tag{3.3}$$

Observe that $D_{KL}(P(\mathbf{U})||P_{JT}(\mathbf{U}))$ follows directly from Equation (2.2); so the term $\sum\limits_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{P_{JT}(\mathbf{U})}{\hat{P}_{JT}(\mathbf{U})} \right)$ remains to be quantified. Using the definition of the probability density function of a junction tree, this can be rewritten as

$$\sum_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{P_{JT}(\mathbf{U})}{\hat{P}_{JT}(\mathbf{U})} \right) = \sum_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{\prod\limits_{C \in \mathcal{C}} p(\mathbf{U}_C)}{\prod\limits_{S \in \mathcal{S}} p(\mathbf{U}_S)} \right)$$
$$- \sum_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{\prod\limits_{C \in \mathcal{C}} \hat{p}(\mathbf{U}_C)}{\prod\limits_{S \in \mathcal{S}} \hat{p}(\mathbf{U}_S)} \right). \tag{3.4}$$

Consider the part of this expression with the estimated terms:

$$- \sum_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{\prod\limits_{C \in \mathcal{C}} \hat{p}(\mathbf{U}_C)}{\prod\limits_{S \in \mathcal{S}} \hat{p}(\mathbf{U}_S)} \right). \tag{3.5}$$

As each random variable appears within a cluster exactly one more time than it appears in a separator [5], adding and subtracting the term
$\sum\limits_{\mathbf{U}} \log \left(\prod\limits_{C \in \mathcal{C}} \hat{p}(U_{C_1}) \cdots \hat{p}(U_{C_k}) \right)$ yields that

$$- \sum_{\mathbf{U}} P(\mathbf{U}) \log \left(\frac{\prod\limits_{C \in \mathcal{C}} \hat{p}(\mathbf{U}_C)}{\prod\limits_{S \in \mathcal{S}} \hat{p}(\mathbf{U}_S)} \right) = \sum_{\mathbf{U}} P(\mathbf{U}) \Bigg[- \log \left(\frac{\prod\limits_{C \in \mathcal{C}} \hat{p}(\mathbf{U}_C)}{\prod\limits_{C \in \mathcal{C}} \hat{p}(U_{C_1}) \cdots \hat{p}(U_{C_k})} \right)$$
$$+ \log \left(\frac{\prod\limits_{S \in \mathcal{S}} \hat{p}(\mathbf{U}_S)}{\prod\limits_{S \in \mathcal{S}} \hat{p}(U_{S_1}) \cdots \hat{p}(U_{S_{k-1}})} \right) - \log \left(\prod_{i=1}^{M} \hat{p}(U_i) \right) \Bigg] \tag{3.6}$$

To simplify notation, for a set of random variables $\mathbf{X} \triangleq \{X_1, \ldots, X_k\}$, define

$$I_p\left(\hat{p}(\mathbf{X})\right) \triangleq \sum_{\mathbf{X}} p(\mathbf{X}) \log \frac{\hat{p}(\mathbf{X})}{\hat{p}(X_1)\cdots\hat{p}(X_k)}. \tag{3.7}$$

Returning to Equation (3.4), it can be rewritten as

$$\sum_{C\in\mathcal{C}}\big[I(p(\mathbf{U}_C)) - I_p(\hat{p}(\mathbf{U}_C))\big] - \sum_{S\in\mathcal{S}}\big[I(p(\mathbf{U}_S)) - I_p(\hat{p}(\mathbf{U}_S))\big]$$
$$- \sum_{i=1}^{M} H(p(U_i)) + \sum_{\mathbf{U}} p(\mathbf{U}) \log\left(\prod_{i=1}^{M}\hat{p}(U_i)\right). \tag{3.8}$$

Note that for every cluster

$$I(p(\mathbf{U}_C)) - I_p(\hat{p}(\mathbf{U}_C)) =$$
$$D_{KL}\left(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)\right) + \sum_{i=1}^{k} D_{KL}(p(U_{C_i})||\hat{p}(U_{C_i})), \tag{3.9}$$

and similarly for separators. Plugging this into Equation (3.8) results in

$$\sum_{C\in\mathcal{C}} D_{KL}(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)) - \sum_{S\in\mathcal{S}} D_{KL}(p(\mathbf{U}_S)||\hat{p}(\mathbf{U}_S)) - \sum_{C\in\mathcal{C}}\sum_{i=1}^{k} D_{KL}(p(U_{C_i})||\hat{p}(U_{C_i}))$$
$$+ \sum_{S\in\mathcal{S}}\sum_{i=1}^{k-1} D_{KL}(p(U_{S_i})||\hat{p}(U_{S_i})) - \sum_{i=1}^{M} H(U_i) + \sum_{\mathbf{U}} p(\mathbf{U}) \log\left(\prod_{i=1}^{M}\hat{p}(U_i)\right). \tag{3.10}$$

Using Lemma 1 from [5] and the fact that

$$\sum_{\mathbf{U}} p(\mathbf{U}) \log\left(\prod_{i=1}^{M}\hat{p}(U_i)\right) = \sum_{i=1}^{M} H(U_i) + D_{KL}(p(U_i)||\hat{p}(U_i)), \tag{3.11}$$

Equation (3.10) becomes

$$\sum_{C\in\mathcal{C}} D_{KL}(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)) - \sum_{S\in\mathcal{S}} D_{KL}(p(\mathbf{U}_S)||\hat{p}(\mathbf{U}_S)). \tag{3.12}$$

Plugging this into Equation (3.3) completes the proof. $\square$

The weight of the estimation terms is defined as w_E and is

$$w_E \triangleq \sum_{C\in\mathcal{C}} D_{KL}(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)) - \sum_{S\in\mathcal{S}} D_{KL}(p(\mathbf{U}_S)||\hat{p}(\mathbf{U}_S)). \tag{3.13}$$

An important property of w_E is that $w_E \geq 0$; that is, using estimated instead of true distributions increases the KL divergence, or at least does not decrease it.

Lemma 3.2.1. *Using distributions estimated from data, as opposed to the true distributions, results in an increase (or at least not a decrease) in KL divergence between the true distribution and the junction tree distribution, i.e.,*

$$\sum_{C \in \mathcal{C}} D_{KL}(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)) - \sum_{S \in \mathcal{S}} D_{KL}(p(\mathbf{U}_S)||\hat{p}(\mathbf{U}_S)) \geq 0. \tag{3.14}$$

Proof. Without loss of generality, assume that for any connected cluster and separator pair, the separator is comprised of $S = \{U_{C_1}, \ldots, U_{C_{k-1}}\} \triangleq \mathbf{U}_{C\setminus k}$. Then utilizing the chain rule of conditional divergence [34],

$$D_{KL}(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)) = D_{KL}\left(p(U_{C_k}|\mathbf{U}_{C\setminus k})||\hat{p}(U_{C_k}|\mathbf{U}_{C\setminus k})\right) + D_{KL}(p(\mathbf{U}_S)||\hat{p}(\mathbf{U}_S)). \tag{3.15}$$

Assume, again without loss of generality, that $\mathcal{C}$ and $\mathcal{S}$ are ordered such that separator S_1 is on the edge between clusters C_1 and C_2, $S_2 = C_2 \cap C_3$, and so forth. Then, as there are $M - k$ cluster-separator pairs and one remaining cluster,

$$\begin{aligned}\sum_{C \in \mathcal{C}} D_{KL}(p(\mathbf{U}_C)||\hat{p}(\mathbf{U}_C)) - \sum_{S \in \mathcal{S}} D_{KL}(p(\mathbf{U}_S)||\hat{p}(\mathbf{U}_S)) = \\ \sum_{C \in \{C_1, \ldots, C_{M-k}\}} D_{KL}\left(p(U_{C_k}|\mathbf{U}_{C\setminus k})||\hat{p}(U_{C_k}|\mathbf{U}_{C\setminus k})\right) \\ + D_{KL}(p(\mathbf{U}_{C_{M-k+1}})||\hat{p}(\mathbf{U}_{C_{M-k+1}})) \geq 0.\end{aligned} \tag{3.16}$$

□

The behavior of the estimation weight w_E is essential to quantify the tradeoff between allowing more complicated correlation structures (thus a more accurate approximation to the true distribution) and the difficulty inherent in estimating distributions that contain more random variables. As the number of entries to be estimated in a distribution scales exponentially with the number of random variables in the joint distribution, an exponential increase in w_E is expected, and Theorem 3.2.2 demonstrates that this is the case.

Theorem 3.2.2. *A) An approximation to the expectation of the KL divergence between a discrete distribution and its estimate, using N samples, is*

$$\mathbb{E}\left[D_{KL}(p(\mathbf{X})||\hat{p}(\mathbf{X}))\right] \sim \frac{1}{2}\sum_{i=1}^{|p|}\frac{1-p_i}{N}. \tag{3.17}$$

B) An approximation to expected value of the estimation error weight for the binary variable case is

$$\mathbb{E}[w_E] \approx \frac{M-k+1}{N}\left(2^{k-1}-2^{-k}\right) - \frac{M-k}{N}\left(2^{k-2}-2^{-(k-1)}\right). \tag{3.18}$$

Proof. A local approximation of the KL divergence between two distributions p and q, $D_{KL}(p||q)$, can be constructed by considering q as a perturbed form of p. Specifically, let $q = p(1+v)$, where v is a vector that redistributes probability mass of p. In order to ensure that $\sum p(1+v) = 1$, it is necessary that $\sum_{i=1}^{|p|} p_i v_i = 0$. This yields the approximation [35]

$$D_{KL}(p||p(1+v)) \sim \frac{1}{2}\sum_{i=1}^{|p|} p_i v_i^2. \tag{3.19}$$

For this proof, consider the divergence of a distribution p over a set of random variables $\mathbf{X} \triangleq \{X_1, \ldots, X_M\}$: $D_{KL}(p(\mathbf{X})||\hat{p}(\mathbf{X}))$. With enough samples N, this divergence should be small, and Equation (3.19) will represent a good approximation to the true divergence. To understand how the divergence between the true distribution and estimated distribution scales as a function of the number of random variables contained in each distribution (the order of the junction tree), consider an entry in the estimated distribution, $\hat{p}_i$. Every $\hat{p}_i$ is considered to be independent of every other $\hat{p}_j$. Each $\hat{p}_i$ is modeled as a binomial random variable with probability of success p_i and N trials. A good approximation when N is large is to consider each $\hat{p}_i$ as as a normal random variable centered at the true value p_i. That is, $\hat{p}_i \sim \mathcal{N}(p_i, \frac{p_i(1-p_i)}{N})$.

Let $Y_i \triangleq v_i\sqrt{p_i}$, then $Y_i \sim \mathcal{N}(0, \frac{1-p_i}{N})$. This implies that $\frac{N}{1-p_i}Y_i^2 \sim \chi_1^2$ and $\mathbb{E}[Y^2] = \frac{1-p_i}{N}$. Plugging this into the expression for the divergence yields

$$\mathbb{E}\left[D_{KL}(p(\mathbf{X})||\hat{p}(\mathbf{X}))\right] \sim \mathbb{E}\left[\frac{1}{2}\sum_{i=1}^{|p|} p_i v_i^2\right] = \mathbb{E}\left[\frac{1}{2}\sum_{i=1}^{|p|} Y_i^2\right] = \frac{1}{2}\sum_{i=1}^{|p|}\frac{1-p_i}{N}, \qquad (3.20)$$

which proves the first part of the theorem.

For the second part of the theorem, a more specific approach is taken, though it should be noted this approach is easily generalized to any collection of discrete random variables. Specifically, consider all random variables X_j, $j = 1, \ldots, M$, to be binary random variables. In order to provide an upper bound on the approximation to the divergence, consider each X_j to be uniformly distributed and assume that this uniformity holds as the number of random variables in the distribution, denoted k, increases. For example, when $k = 1$, $p(\mathbf{X}) = [0.5, 0.5]$ and for $k = 2$, $p(\mathbf{X}) = [0.25, 0.25, 0.25, 0.25]$, and so forth. Note that here these entries in $p(\mathbf{X})$ are in lexicographical order. These assumptions results in the upper bound

$$\mathbb{E}\left[D_{KL}(p(\mathbf{X})||\hat{p}(\mathbf{X}))\right] \approx \frac{1}{N}\left(2^{k-1} - 2^{-k}\right). \qquad (3.21)$$

Using Equation (3.21) and noting that there are $M - k + 1$ clusters and $M - k$ separators completes the proof. □

To demonstrate these concepts, the behavior of the junction tree constructed for hypothesis H_0 as explained in Section 3.3 is presented in Figure 3.1. Note that the approximation from Equation (3.21) captures the scaling behavior of w_E. Additionally, the true junction tree does not exactly model the joint distribution until the order is eight, and thus combined with the increasing w_E, the third order junction tree represents a local minimum of the divergence. Also interesting in this figure is the lack of entries for w_E after the seventh order tree. In this case, there were distributions in the junction tree that were estimated from the training data that resulted

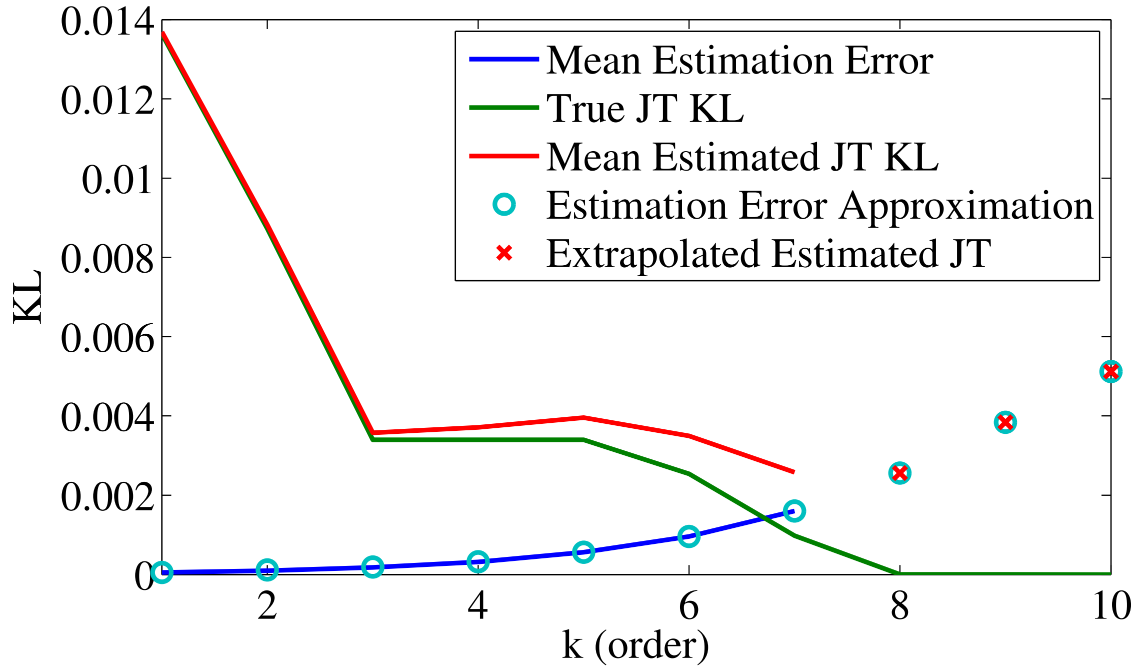


Figure 3.1: Divergence using estimated distributions

in at least one $\hat{p}_i = 0$ where the corresponding $p_i \neq 0$. This represents essentially an absolute continuity error that causes the KL divergence to be ∞. This occurred in an eighth order tree despite having 10^5 samples, which would initially seem adequate to estimate distributions containing eight binary random variables, and thus 256 parameters.

3.3 Numerical Examples

In order to evaluate the approach developed above, utilizing junction trees, as well as examine the impact of constructing a junction tree from training data, we present a simulation study. Consider a sensor deployment of $M = 10$ sensors. The measurement data taken by these sensors is correlated, and the structure of the correlation is captured by a Markov random field. The specific structure of this Markov random field is seen in Figure 3.2. Note that the maximum clique size is

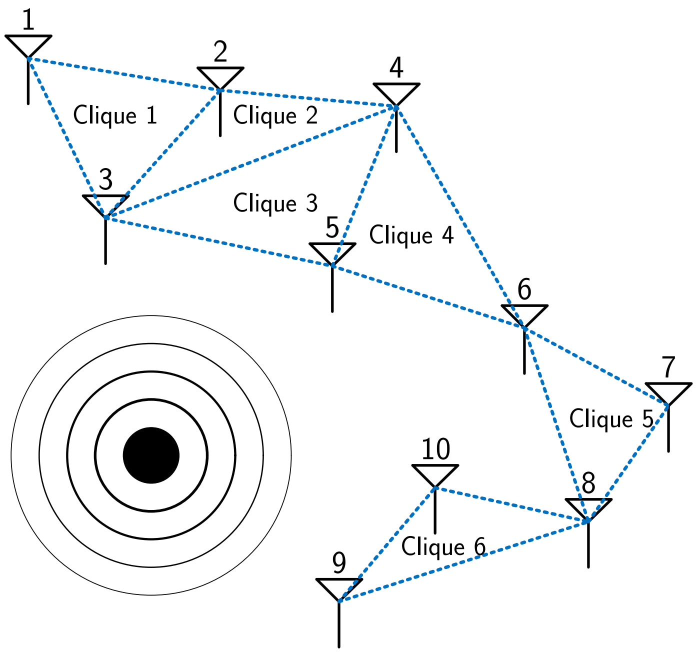


Figure 3.2: Markov random field structure

three, so an optimal t-cherry junction tree of order three will be exactly equal to the true distribution.

For each hypothesis, H_0 and H_1, there are $N = 10^5$ training samples of the true distribution $p(\mathbf{U})$. Using this training data, two t-cherry junction trees are constructed, one for each hypothesis. This process is averaged over 1,000 different realizations of the sample data.

The Receiver Operating Characteristic (ROC) curves of the detector are presented in Figure 3.3 for the cases in which considering the sensors as conditionally independent, second through ninth order t-cherry junction trees, and using the full distribution. Notice that when the sensors are considered conditionally independent, the performance is the worst. Even using a second order junction tree results in a significant improvement. Also, the third through sixth order junction trees all result in essentially identical performance. These ROC curves are not the same as the true

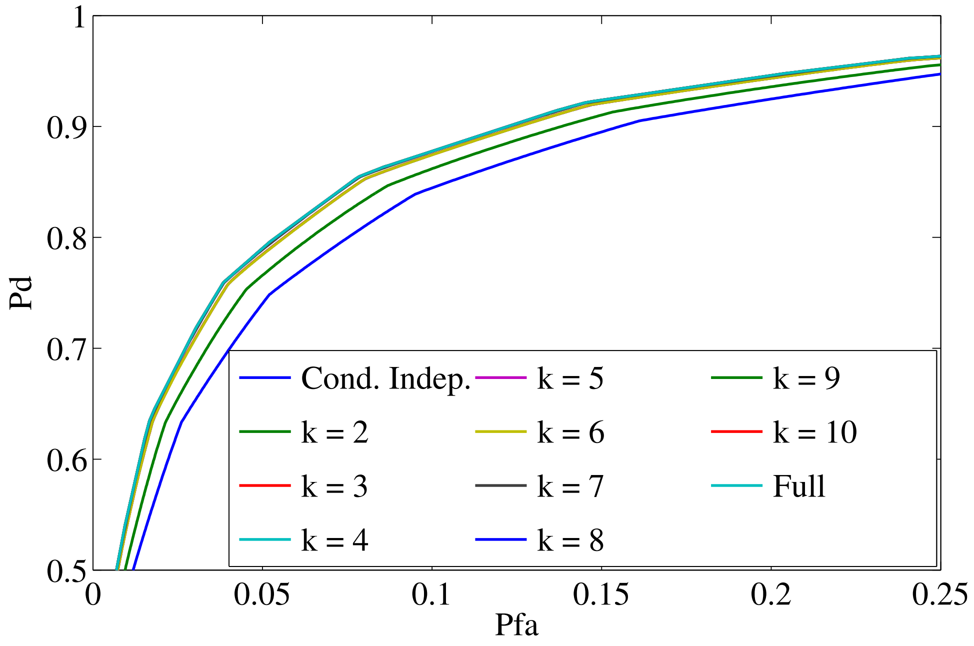


Figure 3.3: ROC curves with true distributions

distribution curve because the t-cherry junction trees constructed were not optimal. From the seventh order onward, the junction trees have managed to exactly replicate the true distribution. In conclusion, using a second and third order t-cherry junction tree offered significant gains over the conditionally independent case, and higher order t-cherry junction trees resulted in diminishing returns.

The area under the ROC curve (AUC) is another metric used to evaluate the performance of a detector. Figure 3.4 plots the performance of t-cherry junction trees when using the true distributions and distributions estimated from training data. Notice that the performances are similar until the ninth order t-cherry junction tree. In this case, the estimated distributions are too inaccurate due to an insufficiency of training data, and the performance is greatly degraded. This emphasizes the importance of correctly trading off the gain resulting from using a (more accurate)

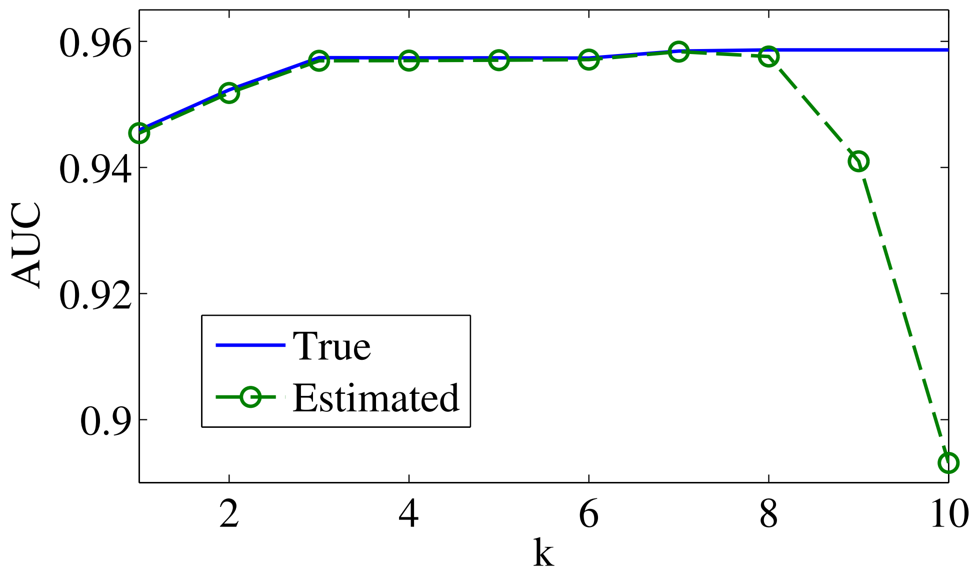


Figure 3.4: AUC of the detectors

higher order t-cherry junction tree against the increased "noise" that results from imprecise estimation of the "larger" distributions involved.

3.4 Conclusion

In this paper, we presented a novel approach to improve the performance of a distributed detection system: the t-cherry junction tree. This tool allows for a smooth tradeoff between treating each sensor as conditionally independent and utilizing the full joint distribution of the individual sensor decisions. Using the t-cherry junction tree allows for some correlation structure in the sensors' decisions to be preserved while simultaneously requiring dramatically fewer parameters to store than storing the full joint distribution.

Additionally, the closed form KL divergence of a junction tree approximation from the true distribution when distributions estimated from data was calculated. This degradation in the ability to closely approximate is exponential in increasing junction tree order, which implies that in many circumstances, there is a maximum order beyond which the error from estimating distributions containing more random variables outpaces any potential gain from allowing more complicated correlation structures.

Chapter 4

SOCIAL TRUST AND SOCIAL RECIPROCITY BASED COOPERATIVE D2D COMMUNICATIONS

4.1 Introduction

Mobile data traffic is predicted to grow further by over 100 times in the next ten years [36], which poses a significant challenge for future cellular networks. One promising approach to increase network capacity is to promote direct communications between hand-held devices. Such device-to-device (D2D) communications can offer a variety of advantages over traditional cellular communications, such as higher user throughput, improved spectral efficiency, and extended network coverage [37]. For example, a device can share the video content with neighboring devices who have the similar watching interest, which can help to reduce the traffic rate demand from the network operator.

Cooperative communication is an efficient D2D communication paradigm where devices can serve as relays for each other [1] . As illustrated in Figure 4.1, cooperative D2D communication can help to 1) improve the quality of D2D communication for direct data offer-loading between devices and 2) enhance the performance of cellular communications between the base station and the devices as well. Hence cooperative D2D communication can be a critical building block for efficient cooperative networking for future wireless networks, wherein individual users cooperate to substantially boost the network capacity and cost-effectively provide rich multimedia services and applications, such as video conferencing and interactive media, anytime, anywhere.

[1]There are many approaches for cooperative communications, and for ease of exposition this study assumes cooperative relaying.

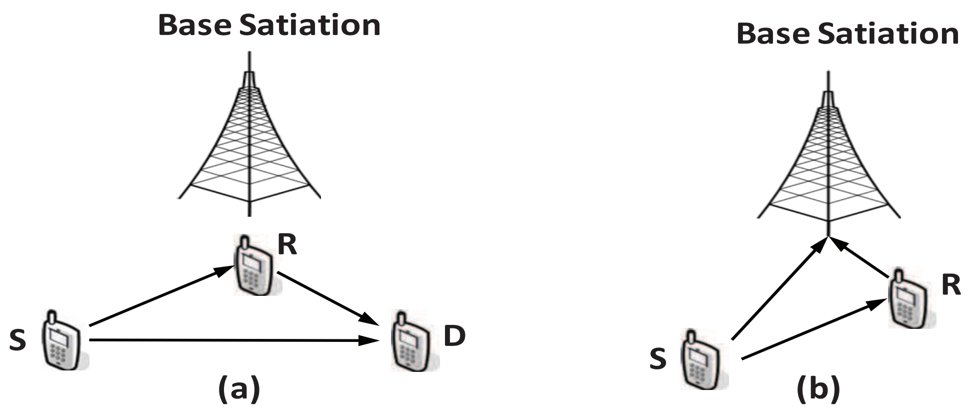


Figure 4.1: An illustration of cooperative D2D communication for cooperative networking. In sub-figure (a), device R serves as the relay for the D2D communication between devices S and D. In sub-figure (b), device R serves as the relay for the cellular communication between device S and the base station. In both cases, the D2D communication between devices S and R is part of cooperative networking.

Nevertheless, a key challenge here is how to stimulate effective cooperation among devices for cooperative D2D communications. As different devices are usually owned by different individuals and they may pursue different interests, there is no good reason to assume that all devices would cooperate with each other.

Since the hand-held devices are carried by human beings, a natural question to ask is that "is it possible to leverage human social relationship to enhance D2D communications for cooperative networking?". Indeed, with the explosive growth of online social networks such as Facebook and Twitter, more and more people are actively involved in online social interactions, and social relationships among people are hence extensively broadened and significantly enhanced [38]. This has opened up a new avenue for cooperative D2D communication system design – we believe that it has potential to propel significant advances in mobile social networking.

One primary goal of this study is to establish a new D2D cooperation paradigm by leveraging two key social phenomena: social trust and social reciprocity. Social trust can be built up among humans such as kinship, friendship, colleague relationship, and altruistic behaviors are observed in many human activities [39]. For example, when a device user is at home or work, typically family members, neighbors, colleagues, or friends are nearby. The device user can then exploit the social trust from these neighboring users to improve the quality of D2D communication, e.g., by asking the best trustworthy device to serve as the relay. Another key social phenomenon, social reciprocity, is also widely observed in human society [40]. Social reciprocity is a powerful social paradigm to promote cooperation so that a group of individuals without social trust can exchange mutually beneficial actions, making all of them better off. For example, when a device user does not have any trusted friends in the vicinity, he (she) may cooperate with the nearby strangers by providing relay assistance for each other to improve the quality of D2D communications.

As illustrated in Figure 4.2, cooperative D2D communications based on social trust and social reciprocity can be projected onto two domains: the physical domain and the social domain. In the physical domain, different devices have different feasible relay selection relationships subject to the physical constraints. In the social domain, different devices have different assistance relationships based on social trust among the devices. In this case, each device has two options for relay selection: 1) either seek relay assistance from another feasible device that has social trust towards him (her); 2) or participate in a group formed based on social reciprocity by exchanging mutually beneficial relay assistance. The main thrust of this study is devoted to tackling two key challenges for the social trust and social reciprocity based approach. The first is which option a device should adopt for relay selection: social trust or social reciprocity. The second is how to efficiently form groups among the devices

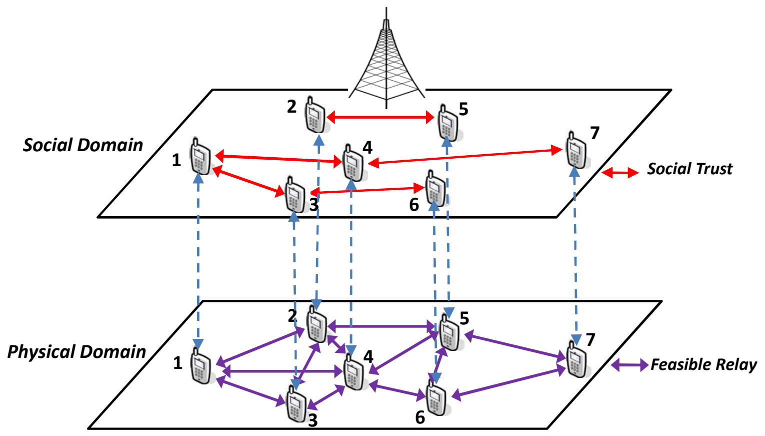


Figure 4.2: An illustration of the social trust model for cooperative D2D communications. In the physical domain, different devices have different feasible cooperation relationships subject to physical constraints. In the social domain, different devices have different assistance relationships based on social trust among the devices.

that adopt the social reciprocity based relay selection. We will develop a coalitional game theoretic framework to address these challenges.

4.1.1 Summary of Main Contributions

The main contributions of this work are as follows:

- *Social trust and social reciprocity based cooperative D2D communications*: We propose a novel social trust and social reciprocity based framework to promote efficient cooperation among devices for cooperative D2D communications. By projecting D2D communications in a mobile social network onto both physical and social domains, we introduce the physical-social graphs to model the interplay therein while capturing the physical constraints for feasible D2D cooperation and the social relationships among devices for effective cooperation.
- *Coalitional game solutions*: We formulate the relay selection problem for social trust and social reciprocity based cooperative D2D communications as a

coalitional game. We show that the coalitional game admits the top-coalition property based on which we devise a core relay selection algorithm for computing the core solution to the game.

- *Network assisted relay selection mechanism*: We develop a network assisted mechanism to implement the coalitional game based solution. We show that the mechanism is immune to group deviations, individually rational, truthful, and computationally efficient. We further evaluate the performance of the mechanism by the real social data trace. Numerical results show that the proposed mechanism can achieve up-to 122% performance gain over the case without D2D cooperation.

A primary goal of this study is to build a theoretically sound and practically relevant framework to understand social trust and social reciprocity based cooperative D2D communications. This framework highlights the interplay between potential physical network performance gain through efficient D2D cooperation and the exploitation of social relationships among device users to stimulate effective cooperation. Besides the cooperative D2D communication scenario where devices serve as relays for each other, the proposed social trust and social reciprocity based framework can also be applied to many other D2D cooperation scenarios, such as cooperative MIMO communications and mobile cloud computing. We believe that these initial steps presented here open a new avenue for mobile social networking and have great potential to enhance network capacity in future wireless networks.

4.1.2 Related Work

Much effort has been made in the literature to stimulate, via incentive mechanisms, cooperation in wireless networks. Payment-based mechanisms have been widely considered to incentivize cooperation for wireless ad hoc networks [8]. Another widely adopted approach for cooperation stimulation is reputation-based mechanisms, where a centralized authority or the whole user population collectively keeps records of the cooperative behaviors and punishes non-cooperating users [9]. However, incentive mechanisms typically assume that all users are fully rational and they act in the selfish manner. Such an assumption are not appropriate for D2D communications as hand-held devices are carried by human beings and people typically act with bounded rationality and involve social interactions [39].

The social aspect is now becoming an important dimension for communication system design. Social structures, such as social community which are derived from the user contact patterns, have been exploited to design efficient data forwarding and routing algorithms in delay tolerant networks [41]. The social influence phenomenon has also been utilized to devise effective data dissemination mechanisms for mobile networks [42]. The common assumption among these works, however, is that all users are always willing to help others, e.g., for data forwarding and relaying. In this work we propose a novel framework to stimulate cooperation among device users while also taking the social aspect into account.

The rest of this chapter is organized as follows. We first introduce the system model in Section 4.2. We then study cooperative D2D communications based on social trust and social reciprocity and develop the network assisted relay selection mechanism in Sections 4.3 and 4.4, respectively. We evaluate the performance of the proposed mechanism by simulations in Section 4.5, and finally conclude in Section 4.6.

4.2 System Model

In this section we present the system model of cooperative D2D communications based on social trust and social reciprocity – a new mobile social networking paradigm. As illustrated in Figure 4.2, cooperative D2D communications can be projected onto two domains: the physical domain and the social domain. In the physical domain, different devices have different feasible cooperation relationships for cooperative D2D communications subject to the physical constraints. In the social domain, different devices have different assistance relationships based on social relationships among the devices. We next discuss both physical and social domains in detail.

4.2.1 Physical (Communication) Graph Model

We consider a set of nodes $\mathcal{N} = \{1, 2, ..., N\}$ where N is the total number of nodes. Each node $n \in \mathcal{N}$ is a wireless device that would like to conduct D2D communication to transmit data packets to its corresponding destination d_n. Notice that a destination d_n may also be a transmit node in the set $\mathcal{N}$ of another D2D communication link or the base station. The D2D communication is underlaid beneath a cellular infrastructure wherein there exists a base station controlling the up-link/down-link communications of the cellular devices. To avoid generating severe interference to the incumbent cellular devices, each node $n \in \mathcal{N}$ will first send a D2D communication establishment request message to the base station. The base station then computes the allowable transmission power level p_n for the D2D communication of node n based on the system parameters such as geolocation of the node n and the protection requirement of the neighboring cellular devices. For example, the proper transmission power p_n of the D2D communication can be computed according to the power control algorithm proposed in [43].

We consider a time division multiple access (TDMA) mechanism in which the transmission time is slotted and one node $n \in \mathcal{N}$ is scheduled to carry out its D2D communication in a time slot [2] . At the allotted time slot, node n can choose either to transmit to the destination node d_n directly or to use cooperative communication by asking another node m in its vicinity to serve as a relay.

Due to the physical constraints such as signal attenuation, only a subset of nodes that are close enough can be feasible relay candidates for the node n. To take such physical constraints into account, we introduce the physical graph [3] $\mathcal{G}^P \triangleq \{\mathcal{N}, \mathcal{E}^P\}$ where the set of nodes $\mathcal{N}$ is the vertex set and $\mathcal{E}^P \triangleq \{(n,m) : e^P_{nm} = 1, \forall n, m \in \mathcal{N}\}$ is the edge set where $e^P_{nm} = 1$ if and only if node m is a feasible relay for node n. An illustration of the physical graph is given in Figure 4.2. We also denote the set of nodes that can serve as a feasible relay of node n as $\mathcal{N}^P_n \triangleq \{m \in \mathcal{N} : e^P_{nm} = 1\}$. A recent work in [44] shows that it is sufficient for a source node to choose the best relay node among multiple candidates to achieve full diversity. For ease of exposition, we hence assume that each node n selects at most one neighboring node $m \in \mathcal{N}^P_n$ as the relay.

For ease of exposition, we consider the full duplex decode-and-forward (DF) relaying scheme [7] for the cooperative D2D communication. Let $r_n \in \mathcal{N}^P_n$ denote the relay node chosen by node $n \in \mathcal{N}$ for cooperative communication. The data rate achieved by node n is then given as [7]

$$Z^{DF}_{n,r_n} = \frac{W}{N} \min\{\log(1 + \mu_{nr_n}), \log(1 + \mu_{nd_n} + \mu_{r_n d_n})\},$$

where W denotes the channel bandwidth and μ_{ij} denotes the signal-to-noise ratio (SNR) at device j when device i transmits a signal to device j. As an alternative, the

[2]Our methods are also applicable to other multiple access schemes.

[3]The graphs (e.g., physical graph and social graph) in this paper can be directed.

node n can also choose to transmit directly without any relay assistance and achieve a data rate of $Z_n^{Dir} = \frac{W}{N}\log(1+\mu_{nd_n})$.

For simplicity, we define the data rate function of node n as $R_n : \mathcal{N}_n^P \cup \{n\} \to \mathbb{R}_+$, which is given by

$$R_n(r_n) = \begin{cases} Z_{n,r_n}^{DF}, & \text{if } r_n \neq n, \\ Z_n^{Dir}, & \text{if } r_n = n. \end{cases} \tag{4.1}$$

We will use the terminology that node n chooses itself as the relay for the situation in which node n transmits directly to its destination d_n.

4.2.2 Social Graph Model

We next introduce the social trust model for cooperative D2D communications. The underlying rationale of using social trust is that the hand-held devices are carried by human beings and the knowledge of human social ties can be utilized to achieve effective and trustworthy relay assistance for cooperative D2D communications.

More specifically, we introduce the social graph $\mathcal{G}^S = \{\mathcal{N}, \mathcal{E}^S\}$ to model the social trust among the nodes. Here the vertex set is the same as the node set $\mathcal{N}$ and the edge set is given as $\mathcal{E}^S = \{(n,m) : e_{nm}^S = 1, \forall n, m \in \mathcal{N}\}$, where $e_{nm}^S = 1$ if and only if nodes n and m have social trust towards each other, which can be kinship, friendship, or colleague relationship between two nodes. We denote the set of nodes that have social trust towards node n as $\mathcal{N}_n^S = \{m : e_{nm}^S = 1, \forall m \in \mathcal{N}\}$, and we assume that the nodes in $\mathcal{N}_n^S$ are willing to serve as the relay of node n for cooperative communication.

Based on the physical graph $\mathcal{G}^P$ and social graph $\mathcal{G}^S$ above, each node $n \in \mathcal{N}$ can classify the set of feasible relay nodes in $\mathcal{N}_n^P$ into two types: nodes with social trust and nodes without social trust. A node n then has two options for relay selection. On the one hand, the node n can choose to seek relay assistance from another feasible device that has social trust towards him (her). On the other hand, the node n can

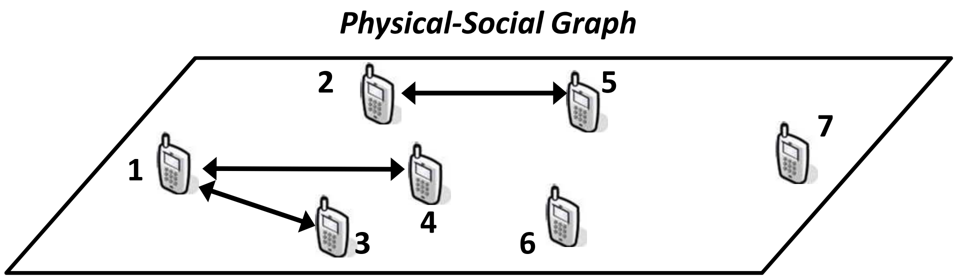


Figure 4.3: The physical-social graph based on the physical graph and social graph in Figure 4.2. For example, there exists an edge between nodes 1 and 3 in the physical-social graph since they can serve as the feasible relay for each other and also have social trust towards each other.

choose to participate in a group formed based on social reciprocity by exchanging mutually beneficial relay assistance. In the following, we will study 1) how to choose between social trust and social reciprocity based relay selections for each node; and 2) how to efficiently form reciprocal groups among the nodes without social trust.

4.3 Social Trust and Social Reciprocity Based Cooperative D2D Communications

In this section, we study the cooperative D2D communications based on social trust and social reciprocity. As mentioned, each node $n \in \mathcal{N}$ has two options for relay selection: social trust based versus social reciprocity. We next address the issues of choosing between social trust and social reciprocity based relay selections for each node and the reciprocal group forming among the nodes without social trust.

4.3.1 Social Trust Based Relay Selection

We first consider social trust based relay selection for D2D cooperation. The key motivation for using social trust is to utilize the knowledge of human social ties to achieve effective and trustworthy relay assistance among the devices for cooperative

D2D communications. For example, when a device user is at home or working place, he (she) typically has family members, neighbors, colleagues, or friends in the vicinity. The device user can then exploit the social trust from neighboring users to improve the quality of D2D communication by asking the best trustworthy device to serve as the relay.

To take both the physical and social constraints into account, we define the *physical-social graph* $\mathcal{G}^{PS} \triangleq \{\mathcal{N}, \mathcal{E}^{PS}\}$ where the vertex set is the node set $\mathcal{N}$ and the edge set $\mathcal{E}^{PS} = \{(n, m) : e^{PS}_{nm} \triangleq e^{P}_{nm} \cdot e^{S}_{nm} = 1, \forall n, m \in \mathcal{N}\}$, where $e^{PS}_{nm} = 1$ if and only if node m is a feasible relay (i.e., $e^{P}_{nm} = 1$) and has social trust towards node n (i.e., $e^{S}_{nm} = 1$). An illustration of the physical-social graph is depicted in Figure 4.3. We also denote the set of nodes that have social trust towards node n and are also feasible relay candidates for node n as $\mathcal{N}^{PS}_{n} = \{m : e^{PS}_{nm} = 1, \forall m \in \mathcal{N}\}$.

For cooperative D2D communications based on social trust, each node $n \in \mathcal{N}$ can choose the best relay to maximize its data rate subject to both physical and social constraints, i.e., $r^{S}_{n} = \arg\max_{r_n \in \mathcal{N}^{PS}_{n} \cup \{n\}} R_n(r_n)$.

4.3.2 Social Reciprocity Based Relay Selection

Next, we study the social reciprocity based relay selection. Different from D2D cooperation based on social trust which requires strong social ties among device users, social reciprocity is a powerful mechanism for promoting mutual beneficial cooperation among the nodes in the absence of social trust. For example, when a device user does not have any friends in the vicinity, he (she) may cooperate with the nearby strangers by providing relay assistance for each other to improve the quality of D2D communications. In general, there are two types of social reciprocity: direct reciprocity and indirect reciprocity [4] (see Figure 4.4 for an illustration). Direct reci-

[4]Reciprocity in this study refers to social reciprocity.

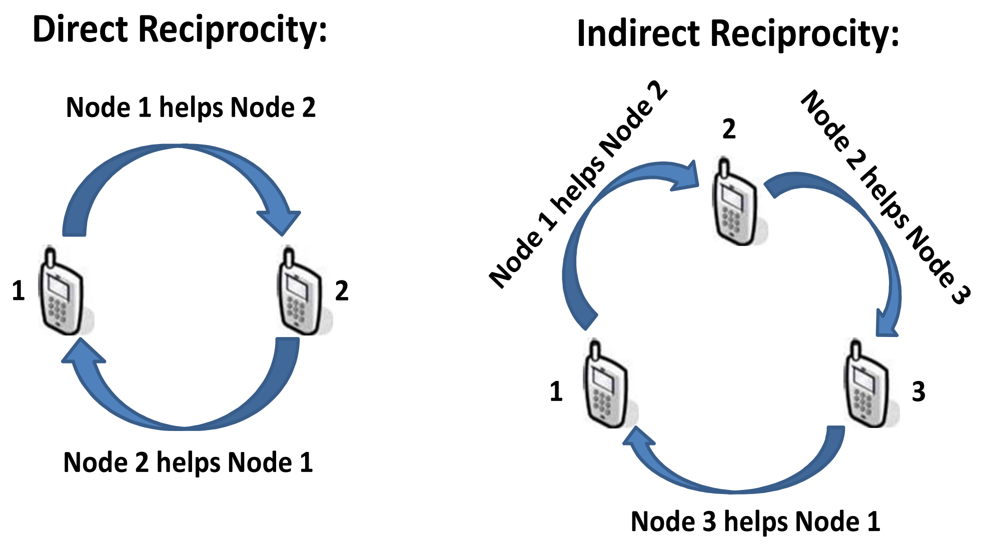


Figure 4.4: An illustration of direct and indirect reciprocity

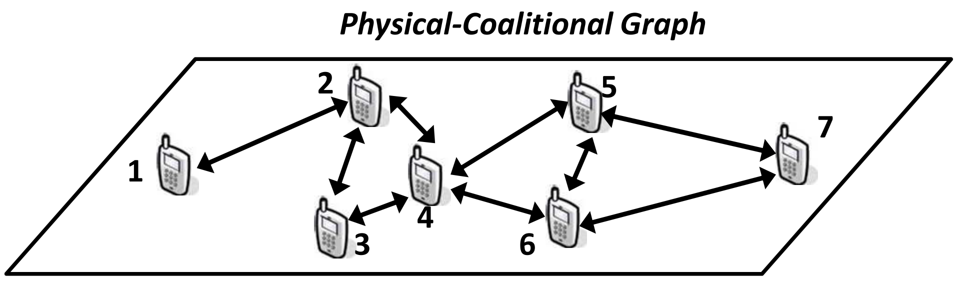


Figure 4.5: The physical-coalitional graph based on the physical graph and social graph in Figure 4.6. For example, there exists an edge between nodes 1 and 2 in the physical-coalitional graph since they can serve as the feasible relay for each other and have no social trust towards each other.

procity is captured in the principle of “you help me, and I will help you”. That is, two individuals exchange altruistic actions so that both obtain a net benefit. Indirect reciprocity is essentially the concept of “I help you, and someone else will help me”. That is, a group of individuals exchange altruistic actions so that all of them can be better off.

To better describe the possible cooperation relationships among the the set of nodes without social trust, we introduce the *physical-coalitional graph* $\mathcal{G}^{PC} = \{\mathcal{N}, \mathcal{E}^{PC}\}$. Here the vertex set is the node set $\mathcal{N}$ and the edge set $\mathcal{E}^{PC} = \{(n,m) : e_{nm}^{PC} \triangleq e_{nm}^{P} \cdot (1 - e_{nm}^{S}) = 1, \forall n, m \in \mathcal{N}\}$, where $e_{nm}^{PC} = 1$ if and only if node m is a feasible relay (i.e., $e_{nm}^{P} = 1$) and has no social trust towards node n (i.e., $e_{nm}^{S} = 0$). An illustration of physical-coalitional graph is depicted in Figure 4.5. We also denote the set of nodes that have no social trust towards user n but are feasible relay candidates of node n as $\mathcal{N}_n^{PC} \triangleq \{m : e_{nm}^{PC} = 1, \forall m \in \mathcal{N}\}$. For social reciprocity based relay selection, a key challenge is how to efficiently divide the nodes into multiple groups such that the nodes can significantly improve their data rates by the reciprocal cooperation within the groups. We next develop a coalitional game framework to address this challenge.

Introduction to the Coalitional Game

For the sake of completeness, we first give a brief introduction to the coalitional game [45]. Formally, a coalitional game consists of a tuple $\Omega = \{\mathcal{N}, \mathcal{X}_{\mathcal{N}}, V, (\succ_n)_{n \in \mathcal{N}}\}$, where

- $\mathcal{N}$ is a finite set of players.
- $\mathcal{X}_{\mathcal{N}}$ is the space of feasible cooperation strategies of all players.
- V is a characteristic function that maps from every nonempty subset of players $\mathcal{S} \subseteq \mathcal{N}$ (a coalition) to a subset of feasible cooperation strategies $V(\mathcal{S}) \subseteq \mathcal{X}_{\mathcal{N}}$. This represents the possible cooperation strategies among the players in the coalition $\mathcal{S}$, given that other players out of the coalition $\mathcal{S}$ do not participate in any cooperation.

- $\succ_n$ is a strict preference order (reflexive, complete, and transitive binary relation) on $\mathcal{X}_{\mathcal{N}}$ for each player $n \in \mathcal{N}$. This captures the idea that different players may have different preferences over different cooperation strategies.

In the same spirit as Nash equilibrium in a non-cooperative game, the "core" plays a critical role in the coalitional game.

Definition 4.3.1. *The core is the set of $\boldsymbol{x} \in V(\mathcal{N})$ for which there does not exist a coalition $\mathcal{S}$ and $\boldsymbol{y} \in V(\mathcal{S})$ such that $\boldsymbol{y} \succ_n \boldsymbol{x}$ for all $n \in \mathcal{S}$.*

Intuitively, the core is a set of cooperation strategies such that no coalition can deviate and improve for all its members by cooperation within the coalition [45].

Coalitional Game Formulation

We then cast the social reciprocity based relay selection problem as a coalitional game $\Omega = \{\mathcal{N}, \mathcal{X}_{\mathcal{N}}, V, (\succ_n)_{n\in\mathcal{N}}\}$ as follows:

- the set of players $\mathcal{N}$ is the set of nodes.
- the set of cooperation strategies $\mathcal{X}_{\mathcal{N}} = \{(r_n)_{n\in\mathcal{N}} : r_n \in \mathcal{N}_n^{PC} \cup \{n\}, \forall n \in \mathcal{N}\}$, which describes the set of possible relay selections for all nodes based on the physical-coalitional graph $\mathcal{G}^{PC}$.
- the characteristic function $V(\mathcal{S}) = \{(r_n)_{n\in\mathcal{N}} \in \mathcal{X}_{\mathcal{N}} : \{r_n\}_{n\in\mathcal{S}} = \{k\}_{k\in\mathcal{S}} \text{ and } r_m = m, \forall m \in \mathcal{N}\backslash\mathcal{S}\}$ for each coalition $\mathcal{S} \subseteq \mathcal{N}$. Here the condition "$\{r_n\}_{n\in\mathcal{S}} = \{k\}_{k\in\mathcal{S}}$" represents the possible relay assistance exchange among the nodes in the coalition $\mathcal{S}$. The condition "$r_m = m, \forall m \in \mathcal{N}\backslash\mathcal{S}$" states that the nodes out of the coalition $\mathcal{S}$ would not participate in any cooperation and choose to transmit directly. For example, in Figure 4.4, the coalition $\mathcal{S} = \{1, 2\}$ in the direct reciprocity case adopts the

cooperation strategy $r_1 = 2$ and $r_2 = 1$ and the coalition $\mathcal{S} = \{1, 2, 3\}$ in the indirect reciprocity case adopts the cooperation strategy $r_1 = 3$, $r_2 = 1$ and $r_3 = 2$.

- the preference order $\succ_n$ is defined as $(r_m)_{m\in\mathcal{N}} \succ_n (r_m^{'})_{m\in\mathcal{N}}$ if and only if $r_n \succ_n r_n^{'}$. That is, node n prefers the relay selection $(r_m)_{m\in\mathcal{N}}$ to another selection $(r_m^{'})_{m\in\mathcal{N}}$ if and only if its assigned relay r_n in the former selection $(r_m)_{m\in\mathcal{N}}$ is better than the assigned relay $r_n^{'}$ in the latter selection $(r_m^{'})_{m\in\mathcal{N}}$. In the following, we define that $r_n \succ_n r_n^{'}$ when $R_n(r_n) > R_n(r_n^{'})$, and if $R_n(r_n) = R_n(r_n^{'})$ then ties are broken arbitrarily.

The core of this coalitional game is a set of $(r_n^*)_{n\in\mathcal{N}} \in V(\mathcal{N})$ for which there does not exist a coalition $\mathcal{S}$ and $(r_n)_{n\in\mathcal{N}} \in V(\mathcal{S})$ such that $(r_n)_{n\in\mathcal{N}} \succ_n (r_n^*)_{n\in\mathcal{N}}$ for all $n \in \mathcal{S}$. In other words, no coalition of nodes can deviate and improve their relay selection by cooperation in the coalition. We will refer the solution $(r_n^*)_{n\in\mathcal{N}}$ as the core relay selection in the sequel.

Core Relay Selection

We now study the existence of the core relay selection. To proceed, we first introduce the following key concepts of coalitional game.

Definition 4.3.2. *Given a coalitional game $\Omega = \{\mathcal{N}, \mathcal{X}_{\mathcal{N}}, V, (\succ_n)_{n\in\mathcal{N}}\}$, we call a coalitional game $\Phi = \{\mathcal{M}, \mathcal{X}_{\mathcal{M}}, V, (\succ_m)_{m\in\mathcal{M}}\}$ a coalitional sub-game of the game Ω if and only if $\mathcal{M} \subseteq \mathcal{N}$ and $\mathcal{M} \neq \varnothing$.*

In other words, a coalitional sub-game Φ is a coalitional game defined on a subset of the players of the original coalitional game Ω.

Definition 4.3.3. *Given a coalitional sub-game $\Phi = \{\mathcal{M}, \mathcal{X}_{\mathcal{M}}, V, (\succ_m)_{m\in\mathcal{M}}\}$, a non-empty subset $\mathcal{S} \subseteq \mathcal{M}$ is a top-coalition of the game Φ if and only if there exists a* *cooperation strategy $(r_m)_{m\in\mathcal{M}} \in V(\mathcal{S})$ such that for any $\mathcal{K} \subseteq \mathcal{M}$ and any cooperation strategy $(r_m^{'})_{m\in\mathcal{M}} \in V(\mathcal{K})$ satisfying $r_m \neq r_m^{'}$ for any $m \in \mathcal{S}$, we have $r_m \succ_m r_m^{'}$ for any $m \in \mathcal{S}$.*

That is, by adopting the cooperation strategy $(r_m)_{m\in\mathcal{S}}$, the coalition $\mathcal{S}$ is a group that is mutually the best for all its members [46].

Definition 4.3.4. *A coalitional game $\Omega = \{\mathcal{N}, \mathcal{X}_{\mathcal{N}}, V, (\succ_n)_{n\in\mathcal{N}}\}$ satisfies the top-coalition property if and only if there exists a top-coalition for any its coalitional sub-game Φ.*

We then show that the proposed coalitional game for social reciprocity based relay selection satisfies the top-coalition property. For simplicity, we first denote $\tilde{\mathcal{N}}_n^{PC} \triangleq \mathcal{N}_n^{PC} \cup \{n\}$. For a coalitional sub-game $\Phi = \{\mathcal{M}, \mathcal{X}_{\mathcal{M}}, V, (\succ_m)_{m\in\mathcal{M}}\}$, we denote the mapping $\gamma(n, \mathcal{M})$ as the most preferable relay of node $n \in \mathcal{M}$ in the set of nodes $\mathcal{M}\cap\tilde{\mathcal{N}}_n^{PC}$, i.e., $\gamma(n, \mathcal{M}) \succ_n i$ for any $i \neq \gamma(n, \mathcal{M})$ and $i \in \mathcal{M}\cap\tilde{\mathcal{N}}_n^{P}$. Based on the mapping γ, we can define the concept of reciprocal relay selection cycle as follows.

Definition 4.3.5. *Given a coalitional sub-game $\Phi = \{\mathcal{M}, \mathcal{X}_{\mathcal{M}}, V, (\succ_m)_{m\in\mathcal{M}}\}$, a node sequence $(n_1, ..., n_L)$ is called a reciprocal relay selection cycle of length L if and only if $\gamma(n_l, \mathcal{M}) = n_{l+1}$ for $l = 1, ..., L-1$ and $\gamma(n_L, \mathcal{M}) = n_1$.*

Notice that when $L = 1$ (i.e., $\gamma(n, \mathcal{M}) = n$), the most preferable choice of node n is to choose to transmit directly; when $L = 2$, this corresponds to the direct reciprocity case; when $L \geq 3$, this corresponds to the indirect reciprocity case. Since the number of nodes (i.e., $|\mathcal{M}|$) is finite, there hence must exist at least one reciprocal relay selection cycle for the coalitional sub-game Φ. This leads to the following result.

Lemma 4.3.1. *Given a coalitional sub-game Φ, there exists at least one reciprocal relay selection cycle. Any reciprocal relay selection cycle is a top-coalition of the coalitional sub-game Φ.*

According to Lemma 4.3.1, we have the following result.

Lemma 4.3.2. *The coalitional game Ω for cooperative D2D communications satisfies the top-coalition property.*

Based on the top-coalition property, we can construct the core relay selection in an iterative manner. Let $\mathcal{M}_t$ denote the set of nodes of the coalitional sub-game $\Phi_t = \{\mathcal{M}_t, \mathcal{X}_{\mathcal{M}_t}, V, (\succ_m)_{m\in\mathcal{M}_t}\}$ in the t-th iteration. Based on the mapping γ and the given set of nodes $\mathcal{M}_t$, we can then find all the reciprocal relay selection cycles as $\mathcal{C}_1^t, ..., \mathcal{C}_{Z_t}^t$ where each cycle $\mathcal{C}_z^t = (n_1^t, ..., n_{|\mathcal{C}_z^t|}^t)$ is a node sequence and Z_t denotes the number of cycles at the t-th iteration. Abusing notation, we will also use $\mathcal{C}_z^t$ to denote the set of nodes in the cycle $\mathcal{C}_z^t$. We can then construct the core relay selection as follows. For the first iteration $t = 1$, we set $\mathcal{M}_1 = \mathcal{N}$ and find the reciprocal relay selection cycles as $\mathcal{C}_1^1, ..., \mathcal{C}_{Z_1}^1$ based on the set of nodes $\mathcal{M}_1$. For the second iteration $t = 2$, we can then set that $\mathcal{M}_2 = \mathcal{M}_1 \backslash \cup_{i=1}^{Z_1} \mathcal{C}_i^1$ (i.e., remove the nodes in the cycles in the previous iteration) and find the new reciprocal relay selection cycles as $\mathcal{C}_1^2, ..., \mathcal{C}_{Z_2}^2$ based on the set of nodes $\mathcal{M}_2$. This procedure repeats until the set of nodes $\mathcal{M}_t = \varnothing$ (i.e., no operation can be further carried out). We summarize the above procedure for constructing the core relay selection in Algorithm 4.

Suppose that the algorithm takes T iterations to converge. We can obtain the set of reciprocal relay selection cycles in all T iterations as $\{\mathcal{C}_i^t : \forall i = 1, ..., Z_t \text{ and } t = 1, ..., T\}$. Since the mapping $\gamma(n, \mathcal{M}_t)$ is unique for each node $n \in \mathcal{M}_t$, we must have that $\cup_{i=1,...,Z_t}^{t=1,...,T} \mathcal{C}_i^t = \mathcal{N}$ (i.e., all the nodes are in the cycles) and $\mathcal{C}_i^t \cap \mathcal{C}_j^{t'} = \varnothing$ for any $i \neq j$ and $t, t' = 1, ..., T$ (i.e., there do not exist any intersecting cycles). For each cycle $\mathcal{C}_i^t = (n_1^t, ..., n_{|\mathcal{C}_i^t|}^t)$, we can then define the relay selection as $r^*_{n_l^t} = n_{l+1}^t$ for any $l = 1, 2..., |\mathcal{C}_i^t| - 1$ and $r^*_{n^t_{|\mathcal{C}_i^t|}} = n_1^t$. We show that $(r_n^*)_{n\in\mathcal{N}}$ is a core relay selection of the coalitional game Ω for the social reciprocity based relay selection.

Theorem 4.3.1. *The relay selection $(r_n^*)_{n\in\mathcal{N}}$ is a core solution to the coalitional game Ω for the social reciprocity based relay selection.*

Proof. We prove the result by contradiction. We assume that there exists a nonempty coalition $\mathcal{S} \subseteq \mathcal{N}$ with another relay selection $(r_m)_{m\in\mathcal{N}} \in V(\mathcal{S})$ satisfying $(r_m)_{m\in\mathcal{N}} \succ_n (r_m^*)_{m\in\mathcal{N}}$ for any $n \in \mathcal{S}$. Let $\mathcal{C}^t = \cup_{i=1}^{Z_t} \mathcal{C}_i^t$ be the set of nodes in the reciprocal relay selection cycles obtained in the t-th iteration. According to Lemma 4.3.1, we know that each cycle $\mathcal{C}_i^1$ is a top-coalition given the set of nodes $\mathcal{M}_1 = \mathcal{N}$. By the definition of top-coalition, we must have that $\mathcal{S} \cap \mathcal{C}^1 = \varnothing$. In this case, we have that $\mathcal{S} \subseteq \mathcal{M}_2 \triangleq \mathcal{M}_1 \backslash \mathcal{C}^1$. Similarly, each cycle $\mathcal{C}_i^2$ is a top-coalition given the set of nodes $\mathcal{M}_2$. We thus also have that $\mathcal{S} \cap \mathcal{C}^2 = \varnothing$. Repeating this argument, we can find that $\mathcal{S} \cap \mathcal{C}^t = \varnothing$ for any $t = 1, ..., T$. Since $\mathcal{N} = \cup_{t=1}^{T} \mathcal{C}^t$, we must have that $\mathcal{S} \cap \mathcal{N} = \varnothing$, which contradicts with the hypothesis that $\mathcal{S} \subseteq \mathcal{N}$ and $\mathcal{S} \neq \varnothing$. This completes the proof. □

Algorithm 4 Core Relay Selection Algorithm

```
1: initialization:
2:     set initial set of nodes $\mathcal{M}_1 = \mathcal{N}$.
3:     set iteration index $t = 1$.
4: end initialization

5: while $\mathcal{M}_t \neq \varnothing$ do
6:     find all the reciprocal relay selection cycles $\mathcal{C}_1^t, ..., \mathcal{C}_{Z_t}^t$.
7:     remove these nodes from the current set of nodes $\mathcal{M}_t$, i.e., $\mathcal{M}_{t+1} = \mathcal{M}_t \backslash \cup_{i=1}^{Z_t} \mathcal{C}_i^t$.

8:     set $t = t + 1$.
9: end while
```

4.3.3 Social Trust and Social Reciprocity Based Relay Selection

According to the principles of social trust and social reciprocity above, each node $n \in \mathcal{M}$ has two options for relay selection. The first option is that node n can choose the best relay $r_n^S = \arg\max_{r_n \in \mathcal{N}_n^{PS} \cup \{n\}} R_n(r_n)$ from the set of nodes with social trust $\mathcal{N}_n^{PS}$. Alternatively, node n can choose a relay $r_n \in \mathcal{N}_n^{PC}$ from the set of nodes without social trust by participating in a directly or indirectly reciprocal cooperation group.

We next address the issue of choosing between social trust and social reciprocity based relay selections for each node, by generalizing the core relay selection $(r_n^*)_{n \in \mathcal{N}}$ in Section 4.3.2. The key idea is to adopt the social trust based relay selection r_n^S as the benchmark for participating in the social reciprocity based relay selection. That is, a node n prefers social reciprocity based relay selection to social trust based relay selection if the social reciprocity based relay selection offers better performance. More specifically, we define that $r_n \succ_n n$ if and only if $r_n \succ_n r_n^S$ and the selection "$r_n = n$" represents that node n will select the relay r_n^S based on social trust. Based on this, we can then compute the core relay selection $(r_n^*)_{n \in \mathcal{N}}$ according to Algorithm 4. In this case, if we have $r_m^* = m$ in the core relay selection $(r_n^*)_{n \in \mathcal{N}}$, then node m will select the relay r_n^S based on social trust. If we have $r_m^* \neq m$ in the core relay selection $(r_n^*)_{n \in \mathcal{N}}$, then node m will select the relay based on social reciprocity.

4.4 Network Assisted Relay Selection Mechanism

In this section, we turn our attention to the implementation of the core relay selection for social trust and social reciprocity based cooperative D2D communications. A key issue here is how to find the reciprocal relay selection cycles in the proposed core relay selection algorithm (see Algorithm 4). In the following, we will first propose an algorithm for finding the reciprocal relay selection cycles, and then develop a network assisted mechanism to implement the core relay selection solution in practical D2D communication systems.

4.4.1 Reciprocal Relay Selection Cycle

We first consider the issue of finding the reciprocal relay selection cycles in the core relay selection algorithm. We introduce a graphical approach to address this issue. More specifically, given the set of nodes $\mathcal{M}_t$ and the mapping γ, we can construct a graph $\mathcal{G}^{\mathcal{M}_t} = \{\mathcal{M}_t, \mathcal{E}^{\mathcal{M}_t}\}$. Here the set of vertices is $\mathcal{M}_t$ and the set of edges $\mathcal{E}^{\mathcal{M}_t} = \{(nm) : e_{nm}^{\mathcal{M}_t} = 1, \forall n, m \in \mathcal{M}_t\}$ where there is an edge directed from node n to m (i.e., $e_{nm}^{\mathcal{M}_t} = 1$) if and only if $\gamma(n, \mathcal{M}_t) = m$.

We next introduce the concept of path in graph theory. A path of length I on a graph is a sequence of nodes $(n_1, n_2, ..., n_I)$ where there is an edge directed from node n_i to n_{i+1} on the graph for any $i = 1, ..., I-1$. A cycle of the graph is a path in which the first and last nodes are identical. A reciprocal relay selection cycle of the coalitional game then corresponds to a cycle of the graph $\mathcal{G}^{\mathcal{M}_t}$. When $\gamma(n, \mathcal{M}_t) = n$, the cycle degenerates to a self-loop of node n. In the following, we say a path $(n_1, n_2, ..., n_I)$ induces a cycle if there exists a path beginning from node n_I that is a cycle. If two cycles are a cyclic permutation of each other, we will regard them as one cycle.

Lemma 4.4.1. *Any sufficiently long path beginning from any node on the graph $\mathcal{G}^{\mathcal{M}_t}$ induces one and only one cycle.*

Based on Lemma 4.4.1, we propose an algorithm to find the reciprocal relay selection cycles in Algorithm 5. The key idea of the algorithm is to explore the paths beginning from each node. More specifically, if a path beginning from a node induces an unfound cycle, then we find a new cycle. We will set the nodes in both the path and cycle as visited nodes since any path beginning from these nodes would induce the same cycle. If a path beginning from a node leads to a visited node, the path would induce a cycle which has already been found if we continue to construct the path on the visited nodes. We will also set the nodes in the path as visited nodes. Since each node will be visited once in the algorithm, the computational complexity of the reciprocal relay selection cycles finding algorithm is $\mathcal{O}(|\mathcal{M}_t|)$.

Algorithm 5 Algorithm For Finding Reciprocal Relay Selection Cycles

```
1: initialization:
2:     construct the graph $\mathcal{G}^{\mathcal{M}_t}$ based on the set of nodes $\mathcal{M}_t$ and the mappings $\{\gamma(n,\mathcal{M}_t)\}_{n\in\mathcal{M}_t}$.
3:     set the set of visited nodes $\mathcal{V}=\varnothing$ and the set of unvisited nodes $\mathcal{U}=\mathcal{M}_t\backslash\mathcal{V}$.
4:     set the set of identified cycles $\triangle=\varnothing$.
5: end initialization

6: while $\mathcal{U}\neq\varnothing$ do
7:    select one node $n_a\in\mathcal{U}$ randomly.
8:    set the set of visited nodes in the current path $\mathcal{H}=\{n_a\}$.
9:    set the flag $F=0$.
10:   while $F\neq 1$: do
11:      generate the next node $n_b=\gamma(n_a,\mathcal{M}_t)$.
12:      if $n_b\in\mathcal{V}$ then
13:         set $\mathcal{V}=\mathcal{V}\cup\mathcal{H}$ and $\mathcal{U}=\mathcal{M}_t\backslash\mathcal{V}$.
14:         set $F=1$.
15:      else if $n_b\in\mathcal{H}$ then
16:         set the identified cycle as $\mathcal{C}=(n_1=n_b,...,n_i=\gamma(n_{i-1},\mathcal{M}_t),...,n_I=n_a)$.
17:         set the set of identified cycles $\triangle=\triangle\cup\{\mathcal{C}\}$.
18:         set $\mathcal{V}=\mathcal{V}\cup\mathcal{H}$ and $\mathcal{U}=\mathcal{M}_t\backslash\mathcal{V}$.
19:         set $F=1$.
20:      else
21:         set $\mathcal{H}=\mathcal{H}\cup\{n_b\}$.
22:         set $n_a=n_b$.
23:      end if
24:   end while
25: end while
```

4.4.2 NARS mechanism

We now propose a network assisted relay selection (NARS) mechanism to implement the core relay selection, which works as follows.

- Each node $n \in \mathcal{N}$ first determines its preference list $\mathcal{L}_n^P$ for the set of feasible relay selections $\tilde{\mathcal{N}}_n^P \triangleq \mathcal{N}_n^P \cup \{n\}$ based on the physical graph $\mathcal{G}^P$. Here $\mathcal{L}_n = (r_n^1, ..., r_n^{|\tilde{\mathcal{N}}_n^P|})$ is a permutation of all the feasible relays in $\tilde{\mathcal{N}}_n^P$ satisfying that $r_n^i \succ_n r_n^{i+1}$ for any $i = 1, ..., |\tilde{\mathcal{N}}_n^P| - 1$. This step can be done through the channel probing procedure to measure the achieved data rate resulting from choosing with different relays.
- Each node $n \in \mathcal{N}$ then computes the best social trust based relay selection $r_n^S = \arg\max_{r_n \in \mathcal{N}_n^{PS} \cup \{n\}} R_n(r_n)$ based on the physical-social graph $\mathcal{G}^{PS}$ and the preference list $\mathcal{L}_n^P$.
- Each node $n \in \mathcal{N}$ next determines its preference list $\mathcal{L}_n^{PC}$ for the set of relay selections $\mathcal{N}_n^{PC} \cup \{n\}$ based on the physical-coalitional graph $\mathcal{G}^{PC}$. Notice that we have that $r_n \succ_n n$ in the preference list $\mathcal{L}_n^{PC}$ if and only if $r_n \succ_n r_n^S$ in the preference list $\mathcal{L}_n^P$.
- Each node $n \in \mathcal{N}$ then reports its preference list $\mathcal{L}_n^{PC}$ to the base-station.
- Based on the preference lists $\mathcal{L}_n^{PC}$ of all nodes, the base-station computes the core relay selection $(r_n^*)_{n \in \mathcal{N}}$ according to Algorithms 4 and 5 and broadcasts the relay selection $(r_n^*)_{n \in \mathcal{N}}$ to all nodes.

As mentioned in Section 4.3.3, if $r_m^* = m$ in the core relay selection $(r_n^*)_{n \in \mathcal{N}}$, then node m will select the relay r_n^S based on social trust. If $r_m^* \neq m$ in the core relay selection $(r_n^*)_{n \in \mathcal{N}}$, then node m will select the relay based on social reciprocity.

Node n	Preference List $\mathcal{L}_n^P$	Relay r_n^S	Preference List $\mathcal{L}_n^{PC}$
1	(1,2,3,4)	1	(1,2)
2	(1,3,2,4,5)	2	(1,3,2,4)
3	(2,3,4,1)	3	(2,3,4)
4	(2,1,4,3,5,6)	1	(2,4,3,5,6)
5	(4,6,7,5,2)	5	(4,6,7,5)
6	(7,5,4,6)	6	(7,5,4,6)
7	(5,6,7)	7	(5,6,7)

Table 4.1: The preference lists of $N = 7$ nodes based on the physical graph $\mathcal{G}^P$ and social graph $\mathcal{G}^S$ in Figure 4.2.

We now use an example to illustrate how the NARS mechanism works. We consider the network of $N = 7$ nodes based on the physical graph $\mathcal{G}^P$ and the social graph $\mathcal{G}^S$ in Figure 4.2. According to NARS mechanism, each node n first determines its preference list $\mathcal{L}_n$ for the set of feasible relay selections $\mathcal{N}_n^P \cup \{n\}$. We will use the preference lists $\mathcal{L}_n^P$ in Table 4.1. For example, in the table the feasible relays for node 7 on the physical graph $\mathcal{G}^P$ are $\{5, 6, 7\}$. The preference list $(5, 6, 7)$ represents that $5 \succ_7 6 \succ_7 \succ 7$, i.e., node 7 prefers choosing node 5 as the relay to choosing node 6 and transmitting directly offers the worst performance. Then based on the physical-social graph $\mathcal{G}^{PS}$ in Figure 4.3 and the preference list $\mathcal{L}_n^P$, each node n computes the best social trust based relay selection r_n^S. For example, node 4's best social trust based relay selection $r_n^S = 1$ (i.e., node 1). Each node n next determines the preference list $\mathcal{L}_n^{PC}$ based on the physical-social graph $\mathcal{G}^{PS}$ in Figure 4.5.

All the nodes then report the preference lists $\mathcal{L}_n^{PC}$ to the base-station. Based on the preference lists, the base-station will compute the core relay selection $(r_n^*)_{n\in\mathcal{N}}$ according to the core relay selection algorithm in Algorithm 4. We illustrate the iterative procedure of the core relay selection algorithm in Figure 4.6 by adopting the graphical representation $\mathcal{G}^{\mathcal{M}_t}$ introduced in Section 4.4.1. Recall that there is an edge directed from node n to node m on graph $\mathcal{G}^{\mathcal{M}_t}$ if node m is the most preferable relay of node n given the set of nodes $\mathcal{M}_t$. At iteration $t = 1$, given that $\mathcal{M}_1 = \mathcal{N}$, the base-station identifies one cycle, i.e., a self-loop formed by node 1. At iteration $t = 2$, given that $\mathcal{M}_2 = \mathcal{M}_1 \backslash \{1\}$, the base-station then identifies one cycle formed by nodes 2 and 3. Notice that graph $\mathcal{G}^{\mathcal{M}_2}$ can be derived from graph $\mathcal{G}^{\mathcal{M}_1}$ by removing node 1 and any edges directed to node 1. For each node (e.g., node 2) from which there is a removed edge directed to node 1, we add a new edge directed from the node to its most preferable node among the set of nodes $\mathcal{M}_2$ (e.g., the edge $2 \rightarrow 3$). We continue in this manner until all the nodes have been removed from the graph. Figure 4.7 shows all the reciprocal relay selection cycles identified by the core relay selection algorithm in Figure 4.6. In this case, the core relay selection is: (a) since $r_1^S = 1$, node 1 transmits directly; (b) nodes 2 and 3 serves as the relay of each other (i.e., direct reciprocity based relay selection); (c) since $r_4^S = 1$, node 4 seeks relay assistance from node 1 (i.e., social trust based relay selection); (d) node 5 serves as the relay of node 7, which in turn serves as the relay of node 6 and node 6 in turn is the relay of node 5 (i.e., indirect reciprocity based relay selection).

4.4.3 Properties of NARS mechanism

We next study the properties of the proposed NARS mechanism. First of all, the following lemma follows from definition of the core solution of a coalitional game.

Lemma 4.4.2. *The core relay selection $(r_n^*)_{n\in\mathcal{N}}$ by NARS mechanism is immune to group deviations, i.e., no group of nodes can deviate and improve by cooperation within the group.*

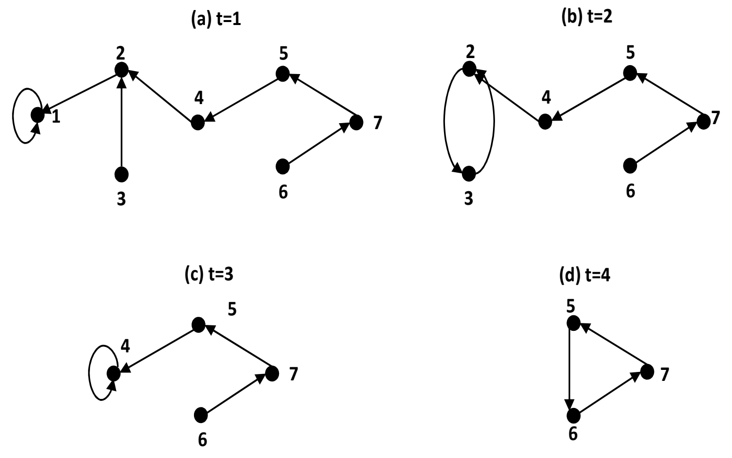


Figure 4.6: An illustration of the resulting graphs $\mathcal{G}^{\mathcal{M}_t}$ at each iteration t of the core relay selection algorithm.

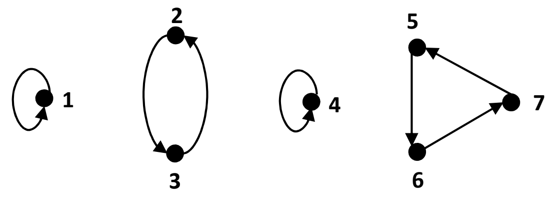


Figure 4.7: The reciprocal relay selection cycles identified by the core relay selection algorithm in Figure 4.6

We can then show that the mechanism guarantees individual rationality, which means that each participating node will not achieve a lower data rate than that when the node does not participate (i.e., in this case the node will transmit directly).

Lemma 4.4.3. *The core relay selection $(r_n^*)_{n\in\mathcal{N}}$ by NARS mechanism is individually rational, i.e., each node $n \in \mathcal{N}$ will be assigned a relay r_n^* which satisfies either $r_n^* \succ_n n$ or $r_n^* = n$.*

Proof. If the assigned relay $r_n^* \prec_n n$ for some node $n \in \mathcal{N}$, then the node n can deviate from the current coalition and improve its data rate by transmitting directly (i.e., $r_n^* = n$). This contradicts with the fact that $(r_n^*)_{n\in\mathcal{N}}$ is a core relay selection. □

We next explore the truthfulness of NARS mechanism. A mechanism is truthful if no node can improve by reporting a preference list different from its true preference list, given that other nodes report truthfully.

Lemma 4.4.4. *NARS mechanism is truthful.*

Proof. Let $\mathcal{C}^t$ be the set of nodes in the reciprocal relay selection cycles obtained in the t-th iteration of core relay selection algorithm. Suppose that the node m reports another preference list that is different from its true preference list. Let τ be the index such that $m \in \mathcal{C}^\tau$. Given that the nodes in the set $\cup_{t=1}^{\tau-1}\mathcal{C}^t$ truthfully report, they will be assigned the relays in the core relay selection regardless of what the nodes out of the set $\cup_{t=1}^{\tau-1}\mathcal{C}^t$ report. In this case, given the set of remaining nodes $\mathcal{M}_\tau = \mathcal{N} \backslash \cup_{t=1}^{\tau-1}\mathcal{C}^t$, the most preferable relay of node m is the relay r_m^* in the core relay selection. This is exactly what the node m achieves by reporting truthfully. Thus, the node m can not improve by reporting another preference list. □

We finally consider the computational complexity of NARS mechanism. We say the mechanism is computationally efficient if the solution can be computed in polynomial time.

Lemma 4.4.5. *The NARS mechanism is computationally efficient.*

Proof. Recall that the reciprocal relay selection cycle finding algorithm in Algorithm 5 has a complexity of $\mathcal{O}(|\mathcal{M}_t|)$. Since the reciprocal relay selection cycle finding algorithm is the dominating step in each iteration, the core relay selection algorithm hence has a complexity of $\mathcal{O}(\sum_{t=1}^{T}|\mathcal{M}_t|)$. As $\sum_{t=1}^{T}|\mathcal{M}_t| = N + \sum_{t=2}^{T}(N - \sum_{\tau=1}^{t-1}|\mathcal{C}^\tau|)$ and $\sum_{t=1}^{T}|\mathcal{C}^\tau| = N$, by setting $|\mathcal{C}^\tau| = 1$ for $\tau = 1, ..., T$, we have the worst case that $\sum_{t=1}^{T}|\mathcal{M}_t| = \sum_{i=1}^{N} i = \frac{N(N+1)}{2}$. Thus, the mechanism has a complexity of at most $\mathcal{O}(N^2)$. □

The above four lemmas together prove the following theorem.

Theorem 4.4.1. *The NARS mechanism is immune to group deviations, individually rational, truthful, and computationally efficient.*

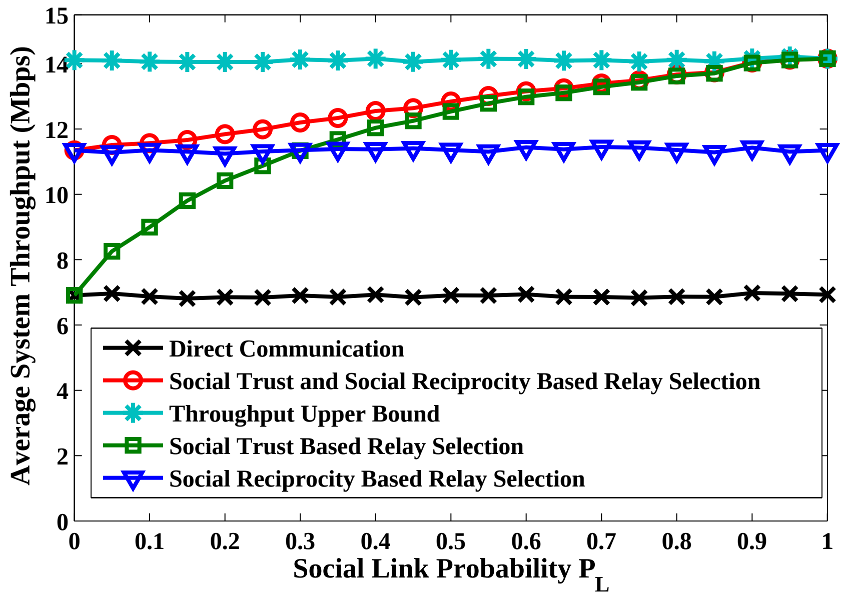
4.5 Simulations

In this section we evaluate the performance of the proposed social trust and social reciprocity based relay selection for cooperative D2D communications through simulations.

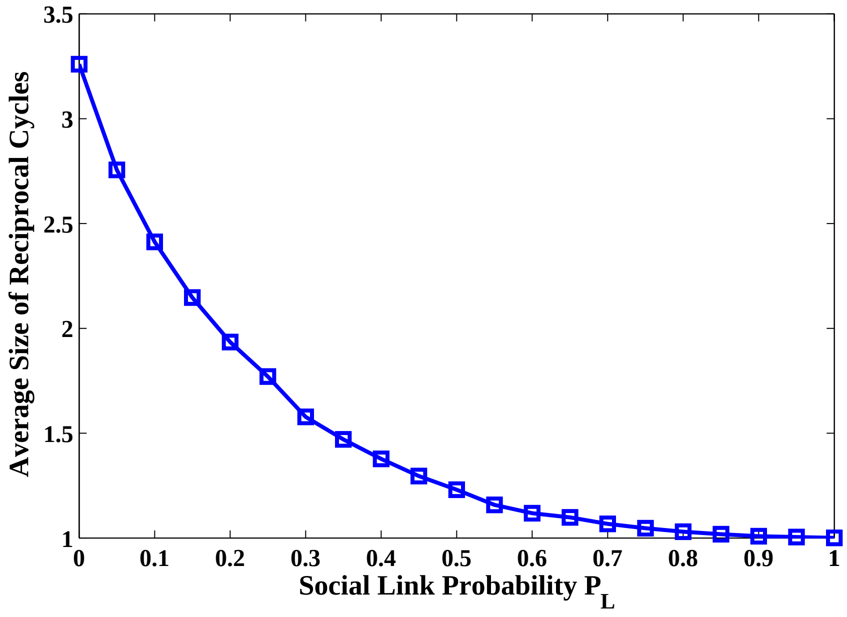
We consider that multiple nodes are randomly scattered across a square area with a side length of 1000 m. Two nodes are randomly matched into a source-destination D2D communication link. We compute the SNR value μ_{ij} according to the physical interference model, i.e., $\mu_{ij} = \frac{p_i}{\omega_0 \cdot ||i,j||^{\alpha}}$ with the transmission power $p_i = 1$ Watt, the background noise $\omega_0 = 10^{-10}$ Watts, and the path loss factor $\alpha = 4$. Based on the SNR μ_{ij}, we set the bandwidth $W = 10$ Mhz and then compute the data rate achieved by using different relays according to Equation (4.1). We construct the physical graph $\mathcal{G}^P$ by setting $e^P_{nm} = 1$ (i.e., node m is a feasible relay of node n) if and only if the distance between nodes n and m is not greater than a threshold $\delta = 500$ m (i.e., $||n,m|| \leq \delta$). For the social trust model, we will consider two types of social graphs: Erdos-Renyi social graph and real data trace based social graph.

4.5.1 Erdos-Renyi Social Graph

We first consider $N = 100$ nodes with the social graph $\mathcal{G}^S$ represented by the Erdos-Renyi (ER) graph model [47] where a social link exists between any two nodes with a probability of P_L. To evaluate the impact of social link density of the social graph, we implement the simulations with different social link probabilities $P_L = 0, 0.05, 0.1, ..., 1.0$, respectively. For each given P_L, we average over 1000 runs. As

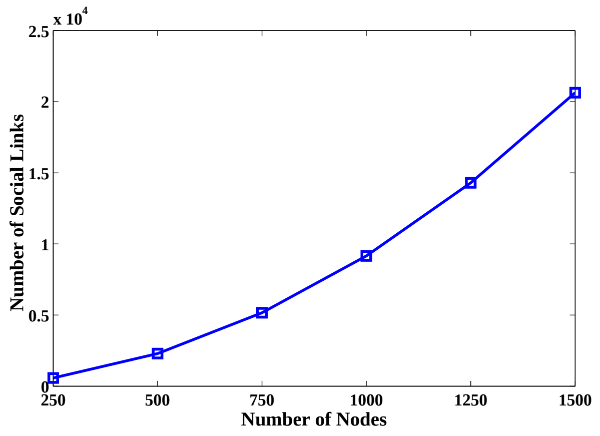


(a) System throughput with the number of nodes $N = 100$ and different social network density.

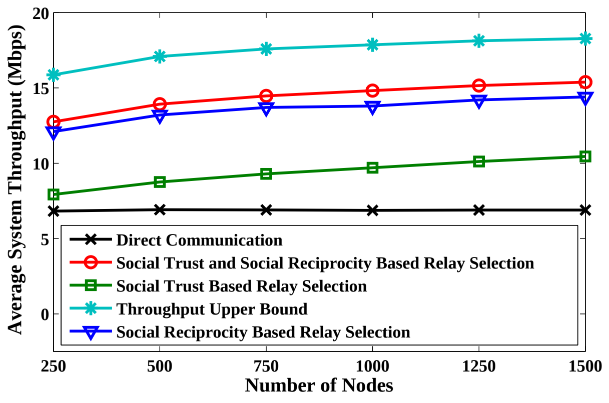


(b) Average size of the reciprocal relay selection cycles in the social trust and social reciprocity based relay selection with $N = 100$ and different social network density.

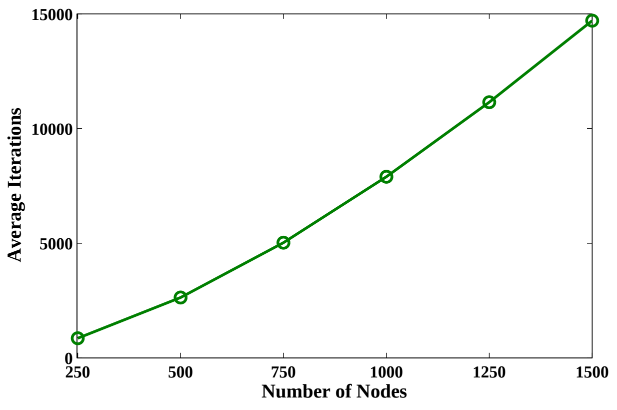
Figure 4.8: Simulation results using an Erdos-Renyi random graph



(a) The number of social links of the social graphs based on real trace Brightkite.



(b) Average system throughput with different number of nodes.



(c) Average number of iterations of the NARS mechanism.

Figure 4.9: Simulation results using the Brightkite social network

the benchmark, we also implement the solution that each node transmits directly, the solution that each node selects the relay based social trust only (i.e., $r_n = r_n^S$), and the solution that each node selects the relay based on social reciprocity only by assuming that there is no social trust among the nodes. Furthermore, we also compute the throughput upper bound by letting each node select the best relay $\bar{r}_n = \arg\max_{r_n \in \mathcal{N}_n^P \cup \{n\}} R_n(r_n)$ among all its feasible relays. Notice that the throughput upper bound can only be achieved when all the nodes are willing to help each other (i.e., all the nodes are cooperative).

We show the average system throughput in Figure 4.8(a). We see that the performance of the social trust and social reciprocity based relay selection dominates that of social trust only based relay selection and social reciprocity only based relay selection. When the social link probability P_L is small, the social trust and social reciprocity based relay selection achieves up to 64.5% performance gain over the social trust only based relay selection. When the social link probability P_L is large, the social trust and social reciprocity based relay selection achieves up to 24% performance gain over the social reciprocity only based relay selection. We also observe that the social trust and social reciprocity based relay selection achieves up-to 100.4% performance gain over the case that all the nodes transmit directly. Compared with the throughput upper bound, the performance loss of the social trust and social reciprocity based relay selection is at most 24%. As the social link probability P_L increases, the social trust and social reciprocity based relay selection improves and approaches the throughput upper bound. This is due to the fact that when the social link probability P_L is large, each node will have a high probability of having social trust from any other node and hence each node is likely to have social trust from its best relay node. This can be illustrated by Figure 4.8(b) that the average size of the reciprocal relay selection cycles in the social trust and social reciprocity based relay selection decreases as the social link probability P_L increases.

4.5.2 Real Trace Based Social Graph

We then evaluate the proposed social trust and social reciprocity based relay selection with the social graphs generated according to the friendship network of the real data trace Brightkite [48]. We implement simulations with the number of nodes $N = 250, 500, ..., 1500$, respectively. The total number of social links among these nodes of the social graphs is shown in Figure 4.9(a).

We show the average system throughput in Figure 4.9(b). We see that the system throughput of the social trust and social reciprocity based relay selection increases as the number of users N increases. This is because that more cooperation opportunities among the nodes are present when the number of users N increases. Moreover, the social trust and social reciprocity based relay selection achieves up-to 122% performance gain over the solution that all users transmit directly. Compared with the throughput upper bound, the performance loss by the social trust and social reciprocity based relay selection is at most 21%. We also show the computational complexity of the NARS mechanism for computing the social trust and social reciprocity based relay selection solution in Figure 4.9(c). We see that the average number of iterations of the mechanism grows linearly as the number of nodes N increases. This demonstrates that the proposed NARS mechanism is computationally efficient (i.e., has a polynomial convergence time).

4.6 Conclusion

In this chapter we studied cooperative D2D communications based on social trust and social reciprocity. We introduced the physical-social graphs to capture the physical constraints for feasible D2D cooperation and the social relationships among devices for effective cooperation. We proposed a coalitional game theoretic approach to find the efficient D2D cooperation strategy and developed a network assisted relay selection mechanism for implementing the coalitional game solution. We showed that the devised mechanism is immune to group deviations, individually rational, truthful, and computationally efficient. We further evaluated the performance of the mechanism based on Erdos-Renyi social graphs and real data trace based social graphs. Numerical results show that the proposed mechanism can achieve up-to 122% performance gain over the case without D2D cooperation.

Chapter 5

AMELIORATING CELLULAR TRAFFIC PEAKS THROUGH PRELOADING AND P2P COMMUNICATIONS

5.1 Introduction

Cellular traffic has been growing exponentially the past several years, including a 230% increase in 2011 [36]. Average smartphone data usage tripled in 2011, and even non-smartphone traffic more than doubled. These sharp increases in mobile traffic are projected to continue over the next several years. In July 2011, Credit Suisse reported that wireless base stations in the United States were operating at 80% of their maximum capacity during busy periods [49]. This combination of exploding demand and limited resources paints a bleak picture of the future for cellular services.

Compounding the issue of congested cellular networks is the impact of rapid interest diffusion in social networks [11] [10]. Content can become viral, in the sense that it has a rapid increase in popularity in a short time, also called a flash crowd [50]. This increase is partly due to online social networks, where users can share their interest in popular content with others. These social networks allow users to interact constantly with each other, rapidly spreading interest in online content. The resulting traffic peaks can be very sharp, for instance in [11], this interest diffusion process has an exponential increase. The traffic spike from a multitude of users becoming interested in the content in a short amount of time is difficult for the cellular network to serve[10].

We propose two key methods to address the problem of the cellular network becoming overloaded during peak times due to interest diffusion in social networks. The first method is via preloading, which enables the cellular network to predict users' demand and preemptively serve some users before they request the content. This allows the cellular network to mitigate spikes in traffic by serving users early, when the network is less congested, without delaying users. The second method is to offload some traffic to a different network, specifically through peer-to-peer (P2P) communications in this study. An alternative method for reducing cellular load spikes is to control the interest diffusion process (cf. [51]) which has been applied to limiting interest diffusion in a social network [52].

Preloading users has also been studied to improve offloading via P2P communications with the goal of minimizing the number of users served by the cellular network. Using *a priori* mobility trace information to determine a set of users to preload in order to maximize the number of users served via P2P communications is studied in [53], and this approach is further expanded to include multiple pieces of content in [54]. The impact of the duration of the meetings between users is quantified in [55]. Rather than selecting the optimal set of preloaded users in advance, other works focus on learning a set over time [56]. Community structure can affect P2P offloading as well [57]. Common to all of these studies is the focus on reducing the number of users served by the cellular network. Although the overall number of users served via cellular is reduced, what remains unclear is when each user is served by the cellular network, and thus cellular traffic peaks may still be high.

Perhaps the model most relevant to ours is [10], in which the authors propose an algorithm that can preload users to minimize the maximum number of users served in a time slot, without delaying any users. It is assumed in [10] that the interest diffusion process is either known *a priori* or can be learned via Markov chain monte carlo methods. In this chapter, we investigate a preloading algorithm that considers both potentially offloaded users and a probabilistic interest diffusion process.

The main thrust of this work is devoted to quantifying the impact of uncertainty in both the inter-meeting process and the interest diffusion process on the peak load of a cellular network. We consider "impatient" users and therefore impose a strict delay constraint on the system: all users must be served before the end of the time slot they become interested. All users are part of a single social network that is modeled as a scale-free random graph. We consider centralized solutions in which a cellular base station can act as the coordinator. We begin by demonstrating that a probabilistic inter-meeting process, as opposed to a process known *a priori*, results in an only marginally increased cellular traffic peak. The increase in the peak load is small enough to be considered negligible, especially considering the significant computational resources required for an algorithm to incorporate a probabilistic process. Based on this insight, we assume a deterministic inter-meeting process for the remainder of the work. Next, we present a greedy preloading algorithm that optimally schedules users one by one and show that this results in the minimum possible cellular traffic peak. This algorithm is optimal for the case when the interest diffusion process is known beforehand and the number of users offloaded is a deterministic function of the preloading schedule.

In contrast to the minimal impact of an uncertain inter-meeting process, we show that the uncertainty in the interest diffusion process, i.e., the uncertainty in the number of newly interested users in each time slot, could result in a significant increase in the cellular traffic peak compared to the peak obtained by our preloading algorithm using an interest diffusion process known *a priori*. The increased peak load is further exacerbated under the condition that not only the number but also the identities of the newly interested nodes are random. Nevertheless, even under these more realistic conditions, utilizing both preloading and offloading offers a significant reduction in the peak compared with doing neither.

Finally, we consider preloading multiple pieces of content to demonstrate the inherent additional difficulties. We illustrate that "naive extensions" from preloading a single piece of content may in fact increase the cellular traffic peak compared to that of not preloading at all. Having shown the importance for new approaches when preloading multiple pieces of content, we develop a heuristic algorithm that offers substantial peak reduction.

The rest of the chapter is organized as follows. We introduce in Section 5.2 the system model and present the interest diffusion process and the content delivery process. In Section 5.3, we study the impact of a random inter-meeting process, and present an algorithm that minimizes the cellular traffic peak assuming perfect information is known. Next, we the investigate the difficulty in selecting which users to preload in Section 5.4. In Section 5.5, we present a new method for preloading users that considers three pieces of content simultaneously, and the chapter concludes in Section 5.6.

5.2 System Model and Problem Formulation

Consider a discrete time system in which each time slot has a fixed duration T. In order to model both the spread of interest in content and the delivery of the content, we investigate two processes. The first process is interest diffusion, by which nodes become interested in a piece of content and seek a copy of it. We assume that the social network is solely responsible for interest diffusion and further assume, for ease of exposition, that there is only a single social network. In this social network, each node represents a user of the cellular network that could become interested in the content. The traffic within the social network is considered insignificant compared to the size of the content and is thus ignored when computing the load of the cellular network. Also, all social network users are assumed to be online during the timescale under consideration.

After becoming interested in a piece of content, a user must then retrieve a copy of it. This retrieval is the second process, which we call content delivery. Content delivery is therefore driven by the interest diffusion process. Each user has two options for obtaining the content, either by using the cellular network or by using P2P communication. As cellular network resources are in high demand, P2P communication is the preferred method for content delivery. However, we assume that P2P communication requires close physical proximity between two users. All users are mobile, and a "meeting" refers to the moment when two users are in range to use P2P communication. We define a user as "served" when that user has obtained a copy of the content and "unserved" otherwise. Both P2P communication and cellular communication are assumed to be instantaneous. Additionally, users are impatient, in the sense that the content must be delivered soon after a user becomes interested in it. Therefore, on the one hand it is beneficial to use P2P communication instead of the cellular network, and on the other hand, users have strict delay constraints that limit the amount of time for an unserved user to meet a served user to use P2P communication.

5.2.1 Interest Diffusion

We begin by studying the interest diffusion of a single piece of content. Consider a given social network consisting of N nodes, each of which corresponds to a user in the cellular network. Interest in the content diffuses through the network as an information cascade [58]: in the first time slot, a node, n_0, is selected uniformly from the social network and becomes interested in the content. At the beginning of the next time slot, each of the neighbors of n_0 becomes interested in the piece of content

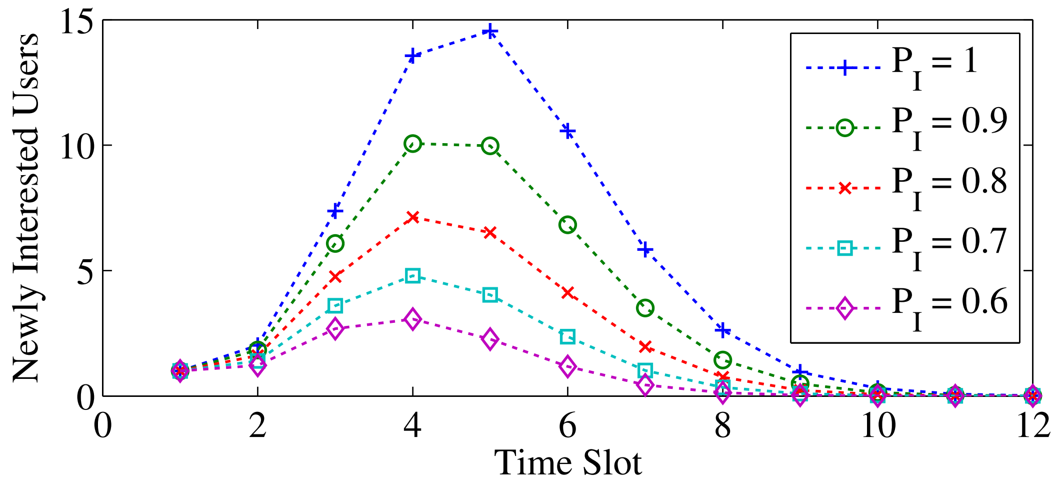


Figure 5.1: Effect of P_I on the interest diffusion process for a 60 node network

with probability P_I. The parameter P_I corresponds to the intuition that not all users are concerned about a given piece of content; rather, some users may find the content worthwhile while others may not. This diffusion process continues until either all nodes are interested or there are no newly interested nodes in a time slot.

The social network is modeled as a scale-free graph [59]. As is standard, we assume that the node degree distribution follows a power law [1] of the form $x^{-\alpha}$, where $2 < \alpha < 3$ [58]. Different P_I representing probabilities of interest can drastically affect the interest diffusion process, which is illustrated in Figure. 5.1. When P_I is low, the overall number of users interested is very low, and there is little benefit to preloading. In order to show the potential gains, we always set P_I close to one in our simulations.

5.2.2 Content Delivery

An interested user can obtain a piece of content using either the cellular network or P2P communication. However, P2P communication can be used only when meeting

[1]For our simulations, the social network is generated by using the Barabasi-Albert algorithm [60]

a user that already has a copy of the content. We assume that all users in the social network access a single cell in the cellular network. If a newly interested node has not met a served node by the end of the time slot, after T seconds, it must retrieve the content using the cellular network. If there is a meeting with a served node, then the content transmission using P2P communication is instantaneous. For tractability, if a newly interested node is served by P2P communications before the end of the time slot, it does not serve other newly interested nodes until the next time slot. Without loss of generality, each user served by the cellular network represents a load of one unit on the cellular network. For example, if five nodes are served by the cellular network in a time slot, then the load of the cellular network is five units in that time slot.

We assume that the inter-meeting process between users is statistically homogeneous and that the inter-meeting time between any two nodes is an exponential random variable with rate λ. Since a newly interested node needs to meet only one served node to obtain a copy of the content, the newly interested node can be served by P2P communications only if the minimum inter-meeting time with a served node is less than the duration of the time slot. Thus the first inter-meeting with a served node is a minimum of exponential random variables, representing the inter-meeting times with served nodes, which is again an exponential random variable.

5.3 Impact of Uncertainty in Inter-Meeting Time and Interest Diffusion

5.3.1 Random Inter-meeting Process

Having defined the interest diffusion and content delivery processes, we begin by studying the problem of minimizing the maximum cellular traffic load in any time slot under strong assumptions. These assumptions provide a baseline to compare against models to be developed later that have fewer assumptions on *a priori* knowledge. We develop a greedy search algorithm and prove it generates a preloading schedule that minimizes the cellular traffic load in any time slot. We will show that the impact of a random inter-meeting process is minimal, and thus use the deterministic process in later sections for ease of exposition.

Consider a known interest diffusion process $\mathcal{D}(k)$ for all time slots $k = 1, \ldots, K$, where $\mathcal{D}(k)$ represents the set of newly interested users in each time slot k. Note that by knowing the identity of the interested nodes, $\mathcal{D}(k)$, the full interest diffusion process is captured by only the number of newly interested users in each time slot, $D(k) \triangleq |\mathcal{D}(k)|$. After K time slots, there are no more newly interested users. We impose a strict delay constraint: a user that becomes interested in time slot k can be served in any earlier time slot $i \leq k$, but no later than time slot k. In order to fully study the impact of preloading, it is assumed that any preloaded user must be served by the end of its preloaded slot. For example, if a node is originally interested in the content in time slot three and is preloaded to be served in time slot two, it must be served by the cellular network at the end of time slot two if it was not served using P2P communication. Though the offloading process is beneficial, the cellular network provider cannot control it. Therefore, preloading is the only controllable aspect to minimize the maximum cellular traffic load in all time slots.

We first evaluate the performance gain through preloading assuming the inter-meeting process is deterministic. Specifically, we assume that the cellular traffic load in a time slot is the expectation of the number of newly interested nodes that do not meet a served node. A new preloading schedule is created, which is denoted as $d(k)$. The cellular load is the number of newly interested nodes served by the cellular network in time slot k. This is a binomial random variable with mean

$$d(k) \cdot \mathbf{exp}\left(-\lambda T \sum_{j=1}^{k-1} d(j)\right). \tag{5.1}$$

The optimal preloading schedule is the solution to the following optimization problem:

$$\begin{aligned} \underset{d(k)}{\text{minimize}} \quad & \max_{k} \; d(k) e^{-\lambda T \sum_{j=1}^{k-1} d(j)} \\ \text{subject to} \quad & \sum_{j=1}^{k} d(j) \geq \sum_{j=1}^{k} D(j), \; k = 1, \ldots, K. \end{aligned} \tag{5.2}$$

Note that the constraint assures that all nodes that become interested during time slot k are served in time slot k or earlier. The cost function is the largest expected value of the cellular load in each time slot.

Based on [10], we next devise a greedy search algorithm to construct a preloading schedule $d(k)$ to solve this integer program. This new algorithm computes an optimal schedule that minimizes the largest cellular traffic load in any time slot. The algorithm places each newly interested user in each time slot one by one, and then schedules the user in the time slot that results in the minimal cost function. To start, $d(1) = D(1)$ in order to satisfy the delay constraints. Next, the first node in $D(2)$ is placed in time slot one and the resulting cost is computed. It is then placed in time slot two, again computing the resulting cost. The node is then scheduled in the time slot with the smallest resulting cost. This process continues for each node in $D(2)$ and then likewise for the following time slots. Note that each node in $D(j)$ can be placed in any of the first j time slots.

Theorem 5.3.1. *With a known interest diffusion process and when the number of users offloaded is a deterministic function of the preloading schedule, the greedy search algorithm presented in Algorithm 6 is optimal, in the sense that it generates a schedule that minimizes the maximum cellular traffic load in any time slot.*

Algorithm 6 Greedy Search Algorithm

```
$d(k) \leftarrow \varnothing\ \forall\, k$
$d(1) \leftarrow D(1)$
for $k = 2 : K$ do
    for all $u \in D(k)$ do
        $k^\star \leftarrow \arg\min_{1 \le j \le k} (d(j)+1) e^{-\lambda T \sum_{i=1}^{j-1} d(i)}$
        $d(k^\star) \leftarrow d(k^\star) \cup u$
    end for
end for
```

Proof. Note that multiple solutions with minimum cost likely exist. We prove the greedy search algorithm returns a schedule with minimum cost by induction. Denote the cost of considering only the first k time slots as $C_G(k)$ for the greedy algorithm and $C_O(k)$ for some unspecified algorithm that returns a schedule with the minimum cost, i.e.,

$$C_O(k) = \min_{d(j)} \max_{1 \le j \le k} d(j) e^{-\lambda T \sum_{i=1}^{j-1} d(i)}. \tag{5.3}$$

Further, as $C_G(k)$ is calculated using the $d(k)$ schedule resulting from Algorithm 6, after k slots,

$$C_G(k) = \max_{1 \le j \le k} d(j) e^{-\lambda T \sum_{i=1}^{j-1} d(i)}. \tag{5.4}$$

Note that $C_G(1) = C_O(1)$ because $d(1) = D(1)$ due to the delay constraint. The time slot that has the maximum cost is defined as k_m, thus

$$d(k_m) e^{-\lambda T \sum_{i=1}^{k_m-1} d(i)} \ge d(k) e^{-\lambda T \sum_{i=1}^{k-1} d(i)}\ \forall\, k. \tag{5.5}$$

In the second time slot if there exists some optimal schedule that results in $C_G(2) > C_O(2)$, then at least one node can be moved that results in a lower cost in time slot k_m. But if one node could be moved that would result in a lower cost, then Algorithm 6 would schedule that node in that time slot. Thus $C_G(2) = C_O(2)$.

Assume $C_G(k-1) = C_O(k-1)$. We prove that our greedy search algorithm results in $C_G(k) = C_O(k)$. There are two possibilities for $C_G(k)$. The first is that $C_G(k) = C_G(k-1)$: the new nodes are scheduled in slots other than k_m. This implies that $C_G(k) = C_O(k)$ because it is impossible to reduce the maximum cost from earlier time slots, i.e., $C_O(k) \geq C_O(k-1)$.

The second possibility is that $C_G(k) > C_G(k-1)$. Within this case, there are two possibilities. The first possibility is that all of the nodes in $D(k)$ are assigned to the last time slot, k. In this case, $k_m = k$. Note that here, $C_G(k)$ depends only on the sum of the previous $k-1$ slots. Therefore, the schedule for all time slots $j < k$ is immaterial. Thus $C_G(k) = C_O(k)$. The other possibility is $k_m < k$. As shown earlier, the greedy search algorithm will return an optimal solution in this case. □

Next, we relax the assumption that the inter-meeting process is deterministic and investigate the impact of uncertainty in this process on the cost. We compare the optimal solution assuming the mean realization from Algorithm 6 with the optimal schedule under the relaxed assumption using a stochastic program. The algorithm for solving the stochastic program is derived from the greedy search algorithm, but the cost is calculated using the entire distribution of the number of meetings between newly interested nodes and served users, rather than calculating the cost by using the expected value. To compare the results of these two approaches, the Value of the Stochastic Solution (VSS) is computed via simulation in a 60 node network.

We study λ in the range from $\frac{1}{200}$ to $\frac{1}{2}$, representing a wide span of inter-meeting probabilities, to explore the possible gain from using the stochastic program. With $\lambda = \frac{1}{2}$, meetings are very likely, thus little preloading is necessary. However, when $\lambda = \frac{1}{200}$, meetings between nodes are rare, and thus many nodes must be preloaded to minimize the peak cellular load. The averaged VSS over all λ is 0.27 and has little deviation over the range of inter-meeting probabilities. Though the stochastic program has better performance, the VSS is small enough to be negligible, indicating that it is not worth the significant additional computation cost.

5.3.2 Random Interest Diffusion

The above study is carried out under the commonly used assumption that the interest diffusion process is completely known *a priori*. Relaxing this strong assumption, we now assume that only the distribution of the number of newly interested users in each time slot is known. Based on the previous section, we assume the inter-meeting process is deterministic because the VSS was small compared to the computational effort required. Therefore, the cost in each time slot is assumed to be the expectation of the number of users served by the cellular network. Since the number of possible D realizations is combinatorial in nature, we rely on simulations to study the impact of a random interest diffusion process. These simulations indicate that a random interest diffusion process greatly degrades the ability to reduce the peak of the cellular traffic load.

We begin by simulating a network containing 60 users. The greedy algorithm presented in Algorithm 6 provides the optimal schedule, denoted d_{det}, by using the mean D realization, which is obtained by averaging over many D realizations. We calculate the average increase in the cost by comparing the cost of using d_{det} to the cost using the optimal schedule assuming D is known using Algorithm 6, for each D realization.

Results from simulations for different λ are presented in Figure 5.2. These are generated by simulating each λ using 50,000 D realizations. The averaged D realization is then used in Algorithm 6 to compute d_{det}. The cost of using d_{det} is compared to the cost of creating a perfect schedule using each D realization and Algorithm 6.

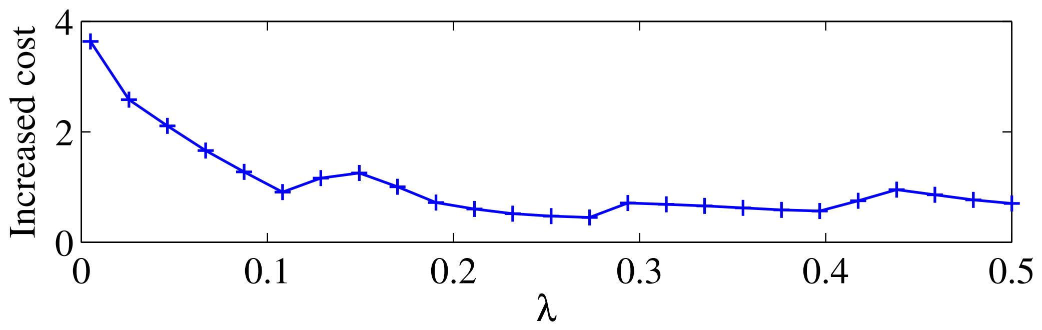


Figure 5.2: Average loss for different inter-meeting probabilities

The average increased cost is high for small λ because there are few inter-meetings, and thus a suboptimal preloading schedule for a D realization incurs a large traffic peak. As the probability of inter-meeting increases, the loss due to imperfect information decreases as suboptimal preloading can be offset by serving a large portion of the nodes via P2P communication.

5.4 Uniform User Selection for Preloading

Next, we consider a more natural model to address the uncertainty in the interest diffusion process. Although in the previous section the *number* of newly interested nodes in each time slot was random, the *identity* of these users was known. To highlight this important difference more concretely, begin by denoting $\mathcal{D}_{\text{det}}(k)$ as the set containing the identities of the nodes in d_{det} that are scheduled in each time slot. For example, let $D = [2, 8, 2]$, with $\mathcal{D}(1) = \{n_a, n_b\}$, $\mathcal{D}(2) = \{n_c, n_d, n_e, n_f, n_g, n_h, n_i, n_j\}$, and $\mathcal{D}(3) = \{n_k, n_l\}$, and assume the optimal preloading schedule is $d_{\text{det}} = [5, 5, 2]$. We assumed that the natural scheduling for d_{det} is $\mathcal{D}_{\text{det}}(1) = \{n_a, n_b, n_c, n_d, n_e\}$, $\mathcal{D}_{\text{det}}(2) = \{n_f, n_g, n_h, n_i, n_j\}$, and $\mathcal{D}_{\text{det}}(3) = \{n_k, n_l\}$. That is, after creating an optimal d_{det}, the user identities were assigned in order from earliest interested to latest. In this section, this strong assumption of knowing the order in which nodes become interested is relaxed, and we turn our attention to the problem of selecting which users to preload.

The identities of the preloaded users are selected uniformly due to our assumption that all nodes are homogeneous. Denote the set of users that are not yet interested or preloaded as $\mathcal{U}$. The greedy search algorithm in Algorithm 6 provides the optimal size of the number of users to preload in each slot, denoted $d^{\star}(k)$, for the original, deterministic interest diffusion process. In time slot one, the original node interested in the file is served by the cellular network, and $d^{\star}(1)-1$ additional users are preloaded. These users are selected uniformly from $\mathcal{U}$. In the next time slot, the interest diffusion process continues using only the original node to spread interest, not any preloaded nodes, as before. If the number of newly interested nodes in time slot two is greater than or equal to $d^{\star}(2)$, then all of these nodes are served. However, if the number of newly interested users is less than $d^{\star}(2)$, users are selected to be preloaded uniformly. This process continues in the remaining time slots.

In Fig. 5.3, we plot the performance of the above approach for a social network with 60 users for different values of λ against the average cost without preloading (using only offloading) and the optimal cost assuming perfect knowledge of $\mathcal{D}(k)$, by averaging over 50,000 different social network realizations. When inter-meetings are rare, there is a large penalty for imperfect user information due to the fact that almost all users are served by the cellular network. However, when nodes are more likely to meet, the loss compared with perfect information is reduced as users are much more likely to be served by P2P communications as opposed to the cellular network. Intuitively, even when selecting users to preload that are not from adjacent time slots, the large number of node inter-meetings mitigates the additional load on the cellular network.

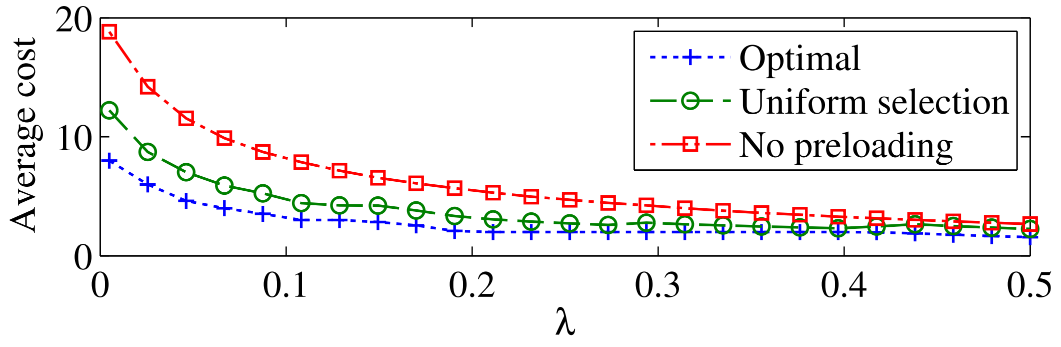


Figure 5.3: Comparison of user selection methods

5.5 The Case with Multiple Pieces of Content

In Sections 5.3.2 and 5.4, we quantified the impact of preloading with limited *a priori* information about the interest diffusion process of a single piece of content. In this section, we study preloading users to deliver multiple pieces of content, as practical cellular networks are expected to serve a multitude of different content to users. In order to highlight the new challenges in preloading multiple pieces of content, we avoid the difficulties that arose during determining which users to preload by reverting to the assumption that the identities of the users are known, i.e., $\mathcal{D}(k)$ is known beforehand for each piece of content. Specifically, we consider three pieces of content, denoted c_1, c_2, c_3, with corresponding interest diffusion processes $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$. These pieces of content do not begin the interest diffusion process at the same time. Instead, each D_i is a time shifted version of the average interest diffusion process in a 60 node network. The interest diffusion process for content c_1 begins in the first time slot. Content c_2 begins the interest diffusion process in time slot three, and c_3 begins in time slot five.

We present three approaches to preload users to serve the content. The first approach is to use Algorithm 6 on D_1 and time shift the resulting d schedule for the other two pieces of content. We denote this method as “restricted preloading” as the later pieces of content are not allowed to be preloaded before their first node becomes interested. The second approach is “single preloading”, in which Algorithm 6 is applied to each D_i independently of the other content, and the resulting d_i schedules are then combined. The later pieces of content, c_2 and c_3 are allowed to preload as early as the first time slot, as opposed to restricted preloading. Neither of these two methods considers the load placed on the cellular network by the other pieces of content when preloading, so the ability to effectively reduce the cellular load peak is reduced.

For the third approach, we present an extension to Algorithm 6 called “aggregate preloading”. The same greedy search algorithm is performed for each content, but in an ordered fashion. For each time slot k, a node from $D_1(k)$ is placed using greedy search first, as D_1 begins earliest. Then a node from $D_2(k)$ is placed using the same greedy search and likewise a node from $D_3(k)$. Unlike the previous two approaches, this greedy search includes the cost information of all nodes already placed for D_1 as well as D_2 and D_3. If a piece of content has already scheduled all of its newly interested nodes from time slot k, then it is skipped.

The averaged performance of these three approaches is compared to the average cost of not preloading in Fig. 5.4. As expected, aggregate preloading outperforms all other methods as it is the only approach that uses the cost information of the other pieces of content to preload.

The performance of the restricted preloading approach highlights a potential problem for preloading. This method represents a naive approach that a practical cellular base station might use for preloading. Each piece of content is treated separately, and future demand for new pieces of content is not known until the first node becomes interested in that content. From Fig. 5.4, this approach has a higher peak load than

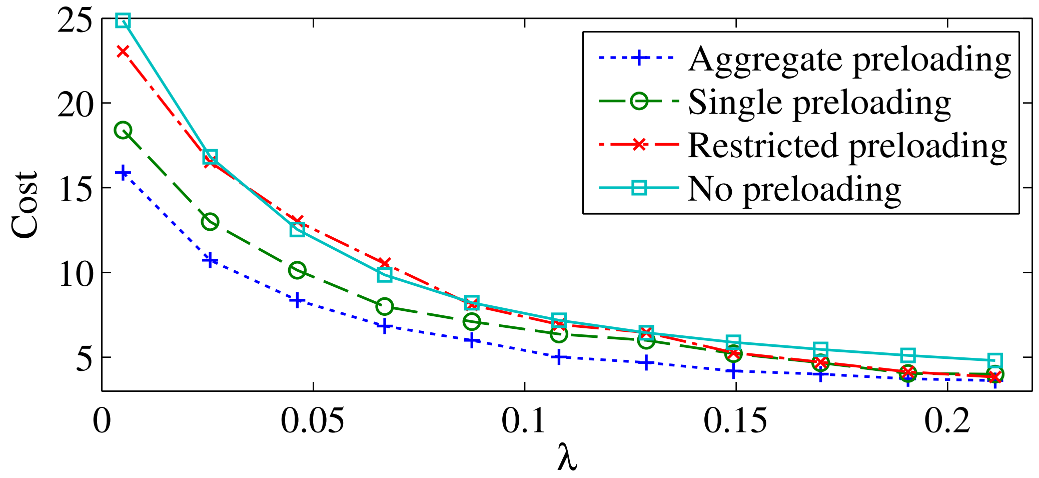


Figure 5.4: Performance of preloading multiple pieces of content

performing no preloading for some inter-meeting probabilities. Thus, careful consideration of the scenarios of interest is needed when considering practical implementations for preloading to avoid accidentally increasing the cellular traffic peak.

5.6 Conclusion

In this chapter, we investigated two methods for reducing the peak of cellular traffic: preloading and offloading. Allowing a cellular network operator to serve users early can mitigate the cellular traffic peak and can be used to create a large base of served users to allow for opportunistically offloading traffic using P2P communication. A greedy preloading algorithm was developed that optimally schedules users when the interest diffusion process is known beforehand, and the impact of scheduling users on P2P communications is a deterministic function. We demonstrated that uncertainty in the inter-meeting process has a minimal impact on the overall peak reduction. In contrast, the uncertainty in the interest diffusion process significantly impairs the ability to minimize the cellular traffic peak. Simulations were used to model the impact of selecting which users to preload on reducing the peak of the cellular load, which illustrated that the cellular traffic peak is still reduced by preloading even under the worst case scenario of homogeneous user identities. Serving multiple pieces of content simultaneously was also studied, and a heuristic algorithm was developed that greatly reduces the cellular peak despite the additional challenges presented when considering multiple pieces of content instead of a single piece.

As the demand for wireless resources continues to increase, solutions to these issues will become increasingly important. More accurate predictions for the interest diffusion process will allow for aggressive preloading that does not risk "mistakenly" serving users that may never become interested in the content.

Chapter 6

CONCLUSION AND FUTURE WORK

In the previous four chapters, a new method to model big data problems by approximating large multivariate distributions, specifically social network data sets, was presented. Then two applications of social network structure were developed: enhancing D2D communications using relays and improving cellular traffic peak performance. Next, these contributions in are summarized in detail.

6.1 Conclusion

We began by modeling social network structure in Chapter 2 by developing a framework based on the t-cherry junction tree to characterize users' relationships in online social networks. To this end, we devised an algorithm to construct a k-order t-cherry junction tree where most of the computations are parallelized. In order to improve the approximation further, we proposed a scheme consisting of the order update and t-cherry conversion steps to construct a higher order t-cherry junction tree. This scheme is significantly faster than building a higher order t-cherry junction tree from scratch and greatly decreases the KL-divergence between the approximation and the joint distribution compared to the original one. This new framework was applied to the new user recommendation problem by creating a probabilistic model of 100,000 user relationships in a Twitter dataset.

This framework was extended in Chapter 3 to include the impact of estimating distributions from a training data set. The loss of fidelity due to imperfect estimates is exactly quantified. Also, the scaling behavior, as the order of the t-cherry junction tree increases, is approximated. This showed an exponential increase in the lost accuracy, which in some circumstances will result in a lower-order tree having a closer approximation to the true high-dimensional distribution. By quantifying the impact of the number of training data samples and the order of the t-cherry junction tree, the tradeoff in higher-order trees can be considered to choose the best possible order for the approximation. These concepts were demonstrated by considering the problem of distributed detection in which the correlation structure in the sensors' measurements is a Markov random field.

The second part of the dissertation focused on applications of this social structure to wireless communication. In Chapter 4, we studied an application involving social networks: cooperative D2D communications based on social trust and social reciprocity. We introduced the physical-social graphs to capture the physical constraints for feasible D2D cooperation and the social relationships among devices for effective cooperation. We then proposed a coalitional game theoretic approach to find the efficient D2D cooperation strategy and developed a network assisted relay selection mechanism for implementing the coalitional game solution. This devised mechanism is immune to group deviations, individually rational, truthful, and computationally efficient. We further evaluated the performance of the mechanism based on Erdos-Renyi social graphs and real data trace based social graphs. Numerical results show that the proposed mechanism can achieve up-to 122% performance gain over the case without D2D cooperation.

Next, we presented a new application leveraging social networks, namely reducing the peak traffic of cellular networks in Chapter 5. Our approach consisted of two methods: preloading and offloading. Allowing a cellular network operator to use predictions of users' content demands to serve users early can mitigate the cellular traffic peak. Additionally, this preloading dovetails with offloading by creating a large base of served users that can opportunistically offload traffic using P2P communication. A greedy preloading algorithm was developed that optimally schedules users under the condition that the interest diffusion process is known beforehand and the impact of scheduling users on P2P communication is a deterministic function. It was demonstrated that uncertainty in the inter-meeting process has a minimal impact on the overall peak reduction. In contrast, uncertainty in the interest diffusion process significantly impairs the ability to minimize the cellular traffic peak. Simulations were used to model the impact of selecting which users to preload on reducing the peak of the cellular load, which illustrated that the cellular traffic peak is still reduced by preloading even under the worst case scenario of homogeneous user identities. Serving multiple pieces of content simultaneously was also studied, and a heuristic algorithm was developed that can reduce the cellular peak despite the additional challenges.

6.2 Future Work

In Chapter 3, it was shown that the KL divergence of the t-cherry junction tree, Equation 3.1, decomposed into two parts: the behavior of the tree with perfect information, w_T, and the impact of using estimated distributions, w_E. The impact of the estimated distributions was quantified in Equation 3.18. However, the other half of the picture, the behavior of the t-cherry junction tree with perfect distribution information was not quantified. Understanding this term would deepen the understanding of the behavior of the KL divergence.

To this end, we will assume that the distribution is a Gauss-Markov Random Field (GMRF) of random variables, in which each random variable is distributed as a Gaussian random variable, with conditional correlation structure as defined by a Markov random field. It is also assumed that every random variable has a mean of zero. This strict assumption is needed as the true t-cherry junction tree behavior is dependent upon the underlying distribution, and without any assumptions, only very general results can be determined.

Specifically, this assumption is that

$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Sigma) \tag{6.1}$$

where $\mathbf{X} \triangleq [X_1, \ldots, X_M]$ and $\mathbf{0}$ is the zero vector of size M.

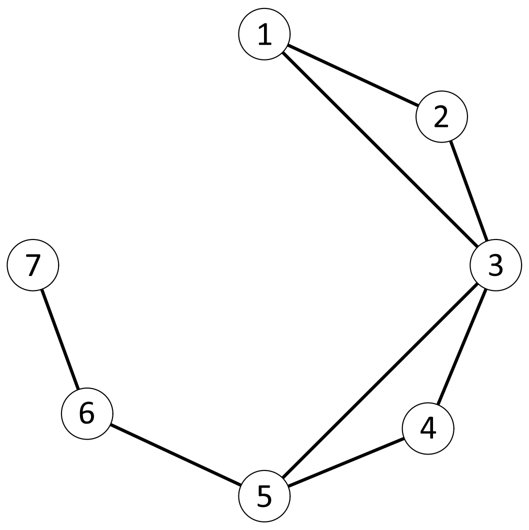
Note that there is a direct connection between a (G)MRF and junction trees. Any chordal MRF induces a junction tree, and a junction tree induces a chordal MRF. This is shown in Figure 6.1. In Figure 6.1(a) is a chordal MRF, and the junction tree it induces is shown in Figure 6.1(b). This junction tree perfectly recreates the distribution over the random variables in the MRF. In Figure 6.1(c), the corresponding t-cherry junction tree is presented. This t-cherry junction tree has the same weight at the junction tree, but retains the t-cherry property.

A known result is that the KL divergence between two multivariate Gaussian distributions containing k random variables, here denoted $N_0 \sim \mathcal{N}(\mathbf{0}, \Sigma_0)$ and $N_1 \sim \mathcal{N}(\mathbf{0}, \Sigma_1)$ with the same mean is

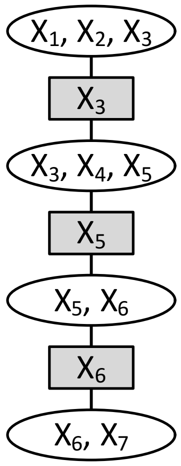
$$D_{KL}(N_0||N_1) = \frac{1}{2}\left(\operatorname{tr}(\Sigma_1^{-1}\Sigma_0) - \log\left(\frac{|\Sigma_0|}{|\Sigma_1|}\right) - k\right), \tag{6.2}$$

where $|\cdot|$ is the determinant of a matrix and $\operatorname{tr}(\cdot)$ is the trace of a matrix. For this work, we are concerned with the case where N_0 represents the true distribution and N_1 represents the situation in which each random variable is considered to be independent of all others. Specifically,

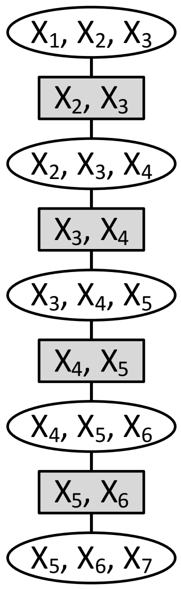
$$\Sigma_0 = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2}^2 & \cdots & \sigma_{1,k}^2 \\ \sigma_{2,1}^2 & \sigma_2^2 & \cdots & \sigma_{2,k}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{k,1}^2 & \sigma_{k,2}^2 & \cdots & \sigma_k^2 \end{bmatrix} \quad (6.3) \qquad \Sigma_1 = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \sigma_k^2 \end{bmatrix}. \quad (6.4)$$



(a) GMRF



(b) Junction tree



(c) T-cherry junction tree

Figure 6.1: GMRF and associated junction trees

Then the KL divergence between the two distributions is

$$D_{KL}(N_0||N_1) = -\frac{1}{2}\log\left(\frac{|\Sigma_0|}{|\Sigma_1|}\right) = -\frac{1}{2}\log\left(\frac{|\Sigma_0|}{\prod_{i=1}^{k}\sigma_i^2}\right). \tag{6.5}$$

The weight of the t-cherry junction tree, w_T, is

$$w_T = \sum_{C\in\mathcal{C}} I(X_C) - \sum_{S\in\mathcal{S}} I(X_S) = -\frac{1}{2}\sum_{i=1}^{M-k+1}\log\left(\frac{|\Sigma_{C_i}|}{\prod_{j=1}^{k}\sigma_j^2}\right) + \frac{1}{2}\sum_{i=1}^{M-k}\log\left(\frac{|\Sigma_{S_i}|}{\prod_{j=1}^{k-1}\sigma_j^2}\right) \tag{6.6}$$

by using Equation 6.5. Note that $\prod_{j=1}^{k}\sigma_j^2$ represents the product of the variances of the random variables in the given cluster, and likewise for a separator.

Using the structure of the t-cherry junction tree,

$$w_T = -\frac{1}{2}\sum_{i=1}^{M-k}\log\left(\frac{|\Sigma_{C_i}|}{|\Sigma_{S_i}|}\right) - \frac{1}{2}\log\left(|\Sigma_{C_{M-k+1}}|\right) + \frac{1}{2}\log\left(\prod_{j=1}^{M}\sigma_j^2\right). \tag{6.7}$$

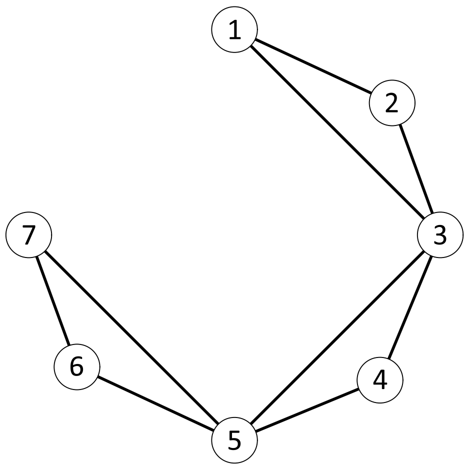


Figure 6.2: Example of a GMRF

The last term, the product of all of the variances, does not depend on the structure of the junction tree.

The precision (information) matrix of a GMRF has a sparse structure based on the conditional correlation graph. If the GMRF contains seven random variables with the dependence structure shown in Figure 6.2, the precision matrix is

$$\Sigma^{-1} = \begin{bmatrix} a & b & b & 0 & 0 & 0 & 0 \\ b & a & b & 0 & 0 & 0 & 0 \\ b & b & a & b & b & 0 & 0 \\ 0 & 0 & b & a & b & 0 & 0 \\ 0 & 0 & b & b & a & b & b \\ 0 & 0 & 0 & 0 & b & a & b \\ 0 & 0 & 0 & 0 & b & b & a \end{bmatrix} \tag{6.8}$$

assuming that the conditional correlations and conditional variances are equal.

In Equation 6.7, the first term can be simplified using the Schur complement, which states that

$$|\Sigma_C| = |\Sigma_S|(\sigma_k^2 - v^T \Sigma_S^{-1} v), \tag{6.9}$$

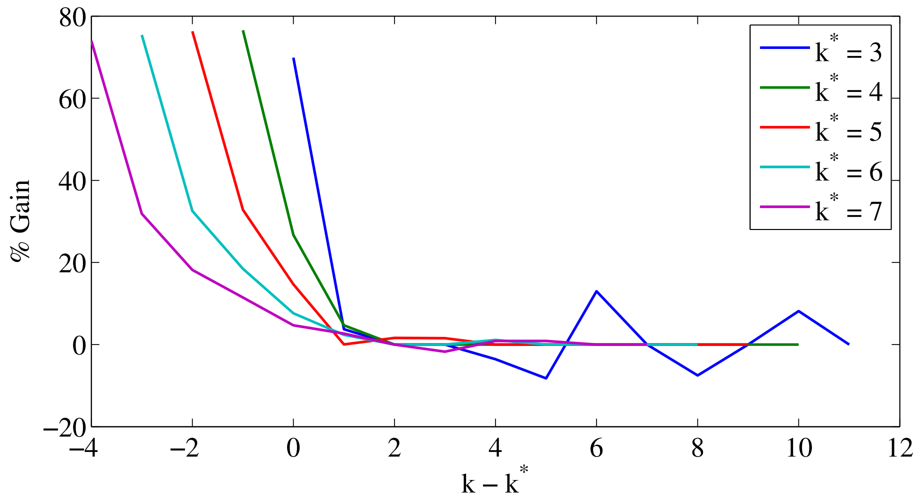


Figure 6.3: Greedy algorithm performance for GRMFs

where v is a vector containing the correlations between the dominating vertex of the cluster and the variables in the separator.

As the order of the junction tree increases, there are fewer clusters and separators, however we expect the weight term w_T to improve only marginally after the true order (maximal clique size of the MRF) of the underlying distribution (which is three for the GMRF in Figure 6.2).

As higher-order junction trees are considered, and w_T saturates, most of this saturation term comes from the middle term of Equation 6.7,

$$-\frac{1}{2}\log\left(|\Sigma_{C_{M-k+1}}|\right). \tag{6.10}$$

Obviously, when the junction tree order is the same as M, there is only one cluster and all of the weight of the three is contained in this term. With two clusters, i.e., $k = M-1$, most of the weight is contained in this term as opposed to the cancelation of the other cluster and the one separator.

Despite the difficulties in analytically proving these points, we have a strong conjecture. This conjecture is that the weight of the t-cherry junction tree constructed using the greedy algorithm, Algorithm 1 in Chapter 2, saturates after the order of the t-cherry junction tree exceeds the order of the underlying GMRF (here denoted k^*.) To support this conjecture, simulation studies involving synthetic GMRFs were performed, the results of which can be seen in Figure 6.3. In this figure, the x-axis represents the order of the t-cherry junction minus the true order of the underlying GMRF. For example, for the purple curve, $k^* = 7$, the weight of the 3-order t-cherry junction tree is plotted on the x-axis at -4, as $3 - 7 = -4$. The y-axis represents the percent gain in the weight of the k-order t-cherry junction tree compared to the previous, (k-1)-order tree. Note that beyond $k - k^* = 1$, the percent gains quickly converge to zero. Note that were the optimal t-cherry junction trees constructed, these weights would be zero beyond $k = k^*$. Using the connection between constructing a t-cherry junction tree and inducing a MRF structure, the greedy algorithm fails to link certain variables that are linked in the underlying distribution, but as the order increases, these missing links are added.

In conclusion, this ongoing work has the potential to complete the storyline of the t-cherry junction tree. In practice, the fundamental question for using this framework is: what is the correct order of t-cherry junction tree to build? That is, how to trade-off between the complexity in constructing one and the increased storage and training data requirements against the increasing fidelity (weight) of the tree. The error due to estimating distributions from data was shown to be exponential in order, and thus beyond some order, this w_E term will greatly increase the KL divergence. However, the behavior of the weight of the true tree, w_T, is conjectured to quickly saturate past the clique size of the underlying distribution. Thus, there should be a point at which the KL divergence of the approximation to the true high-dimensional joint distribution begins to increase quickly, and this point represents a good heuristic for choosing the best order for the t-cherry junction tree.

REFERENCES

[1] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, ser. CIKM '03. New York, NY, USA: ACM, 2003, pp. 556–559.

[2] H. Zhuang, J. Tang, W. Tang, T. Lou, A. Chin, and X. Wang, "Actively learning to infer social ties," *Data Mining and Knowledge Discovery*, vol. 25, no. 2, pp. 270–297, 2012.

[3] M. Rowe, M. Stankovic, and H. Alani, "Who will follow whom? Exploiting semantics for link prediction in attention-information networks," in *Proceedings of the 11th international conference on The Semantic Web - Volume Part I*, ser. ISWC'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 476–491.

[4] D. Koller and N. Friedman, *Probabilistic Graphical Models - Principles and Techniques.* MIT Press, 2009.

[5] T. Szantai and E. Kovacs, "Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution," *Annals of Operations Research*, vol. 193, no. 1, pp. 71–90, 2012.

[6] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[7] A. Host-Madsen and J. Zhang, "Capacity bounds and power allocation for wireless relay channels," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 2020–2040, June 2005.

[8] L. Anderegg and S. Eidenbenz, "Ad hoc-VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '03. New York, NY, USA: ACM, 2003, pp. 245–259.

[9] P. Michiardi and R. Molva, "CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security.* Deventer, The Netherlands, The Netherlands: Kluwer, B.V., 2002, pp. 107–121.

[10] F. Malandrino, M. Kurant, A. Markopoulou, C. Westphal, and U. C. Kozat, "Proactive seeding for information cascades in cellular networks," in *Proc. INFOCOM '12*, 2012, pp. 1719 –1727.

[11] S. Shakkottai and R. Johari, "Demand-aware content distribution on the Internet," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 476 –489, april 2010.

[12] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *In Proceedings of Uncertainty in AI*, 1999, pp. 467–475.

[13] N. Santhanam and M. Wainwright, "Information-theoretic limits of selecting binary graphical models in high dimensions," *Information Theory, IEEE Transactions on*, vol. 58, no. 7, pp. 4117–4134, July 2012.

[14] G. Webb, J. Boughton, F. Zheng, K. Ting, and H. Salem, "Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification," *Machine Learning*, vol. 86, no. 2, pp. 233–272, 2012.

[15] (2013, May) Facebook reports first quarter 2013 results. [Online]. Available: http://investor.fb.com/releasedetail.cfm?ReleaseID=761090

[16] (2013, May) Twitter statistics. [Online]. Available: http://www.statisticbrain.com/twitter-statistics/

[17] E. Kovacs and T. Szantai, "On the approximation of a discrete multivariate probability distribution using the new concept of t-cherry junction tree," *Lecture Notes in Economics and Mathematical Systems*, vol. 633, pp. 39–56, 2010.

[18] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Dec. 1998.

[19] J. Bukszar and A. Prekopa, "Probability bounds with cherry trees," *Mathematics of Operations Research*, vol. 26, no. 1, pp. pp. 174–192, 2001.

[20] J. Bukszar, "Upper bounds for the probability of a union by multitrees," *Advances in Applied Probability*, vol. 33, no. 2, pp. pp. 437–452, 2001.

[21] T. Szantai and E. Kovacs, "Discovering a junction tree behind a Markov network by a greedy algorithm," *Optimization and Engineering*, vol. 14, pp. 503–518, 2013.

[22] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, May 1968.

[23] A. Deshpande, M. Garofalakis, and M. Jordan, "Efficient stepwise selection in decomposable models," in *In Proc. UAI.* Morgan Kaufmann Publishers, 2001, pp. 128–135.

[24] F. Malvestuto, "A backward selection procedure for approximating a discrete probability distribution by decomposable models," in *In Proc. Kybernetika*, 2012, pp. 825–844.

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

[26] A. Iosup, S. Ostermann, M. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 931–945, 2011.

[27] (2013, July) Metis 5.1.0 software. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/metis/metis/download

[28] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[29] Y. Xia and V. Prasanna, "Distributed evidence propagation in junction trees on clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 7, pp. 1169–1177, 2012.

[30] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Graphlab: A new parallel framework for machine learning," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July 2010.

[31] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, November 1984.

[32] ——, "Decentralized detection by a large number of sensors," *Mathematics of Control, Signals, and Systems*, 1988.

[33] C. W. Helstrom, *Elements of Signal Detection and Estimation.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[34] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, 2006.

[35] E. Abbe and L. Zheng, "Linear universal decoding for compound channels," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 5999–6013, Dec 2010.

[36] "Cisco visual networking index: Global mobile data traffic forecast update, 2011-2016," White Paper, Cisco, February 2012.

[37] G. Fodor, E. Dahlman, G. Mildh, S. Parkvall, N. Reider, G. Miklos, and Z. Turanyi, "Design aspects of network assisted device-to-device communications," *IEEE Communications Magazine*, vol. 50, no. 3, pp. 170–177, March 2012.

[38] N. Kayastha, D. Niyato, P. Wang, and E. Hossain, "Applications, architectures, and protocol design issues for mobile social networks: A survey," *Proceedings of the IEEE*, vol. 99, no. 12, pp. 2130–2158, Dec 2011.

[39] T. Govier, *Social Trust and Human Communities.* McGill-Queen's University Press, 1997.

[40] H. Gintis, “Strong reciprocity and human sociality,” *Journal of Theoretical Biology*, vol. 206, no. 2, pp. 169 – 179, 2000.

[41] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco, “Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 5, pp. 748–760, Jun. 2008.

[42] C. Boldrini, M. Conti, and A. Passarella, “Contentplace: Social-aware data dissemination in opportunistic networks,” in *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM ’08. New York, NY, USA: ACM, 2008, pp. 203–210.

[43] C.-H. Yu, O. Tirkkonen, K. Doppler, and C. Ribeiro, “On the performance of device-to-device underlay communication with simple power control,” in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, April 2009, pp. 1–5.

[44] Y. Zhao, R. Adve, and T. J. Lim, “Improving amplify-and-forward relay networks: optimal power allocation versus selection,” in *IEEE International Symposium on Information Theory.* IEEE, 2006, pp. 1234–1238.

[45] R. Myerson, *Game theory: analysis of conflict.* Harvard University Press, 1991.

[46] S. Banerjee, H. Konishi, and T. Snmez, “Core in a simple coalition formation game,” *Social Choice and Welfare*, vol. 18, no. 1, pp. 135–153, 2001. [Online]. Available: http://dx.doi.org/10.1007/s003550000067

[47] M. E. Newman, D. J. Watts, and S. H. Strogatz, “Random graph models of social networks,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. Suppl 1, pp. 2566–2572, 2002.

[48] J. Leskovec, “Brightkite dataset,” 2012. [Online]. Available: http://snap.stanford.edu/data/loc-brightkite.html

[49] P. Goldstein. (2012, June) Credit Suisse report: U.S. wireless networks running at 80% of total capacity. [Online]. Available: http://benton.org/node/81874

[50] J. Jung, B. Krishnamurthy, and M. Rabinovich, “Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites,” in *Proc. WWW ’02.* New York, NY, USA: ACM, 2002, pp. 293–304.

[51] D. Kempe, J. Kleinberg, and E. Tardos, “Maximizing the spread of influence through a social network,” in *Proc. KDD ’03*, 2003, pp. 137–146.

[52] H. Sharara, C. Westphal, S. Radosavac, and U. Kozat, “Utilizing social influence in content distribution networks,” in *Proc. IEEE ICC ’11*, 2011, pp. 1 –6.

[53] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, “Cellular traffic offloading through opportunistic communications: a case study,” in *Proc. ACM CHANTS ’10*, 2010, pp. 31–38.

[54] Y. Li, G. Su, P. Hui, D. Jin, L. Su, and L. Zeng, “Multiple mobile data offloading through delay tolerant networks,” in *Proc. ACM CHANTS ’11*, 2011, pp. 43–48.

[55] X. Zhuo, Q. Li, W. Gao, G. Cao, and Y. Dai, “Contact duration aware data replication in delay tolerant networks,” in *Proc. ICNP ’11*, 2011, pp. 236 –245.

[56] M. Barbera, J. Stefa, A. Viana, M. de Amorim, and M. Boc, “VIP delegation: enabling VIPs to offload data in wireless social mobile networks,” in *Proc. DCOSS 2011*, 2011, pp. 1 –8.

[57] A.-K. Pietilänen and C. Diot, “Dissemination in opportunistic social networks: the role of temporal communities,” in *Proc. MobiHoc ’12*, 2012, pp. 165–174.

[58] A. Barrat, M. Barthelemy, and A. Vespignani, *Dynamical Processes on Complex Networks.* Cambridge University Press, 2008.

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

[60] R. Albert and A.-L. Barabási, “Statistical mechanics of complex networks,” *Reviews of Modern Physics*, vol. 74, pp. 47–97, 2002.