Network Topology Tomography

Aaron James Defazio

A thesis submitted in partial fulfillment of the degree of
Bachelor of Computer Science (Honours) at
The Department of Computer Science
Australian National University

October 2010

Except where otherwise indicated, this thesis is my own original work.

Aaron James Defazio

26 October 2010

Abstract

In this work we consider the problem of reconstructing the topology of networks from indirect measurements. A novel algorithm is developed for reconstruction from coocurrence samples—sets of nodes that are known to form a path in the network, but for which the order is unknown. This algorithm outperforms existing algorithms for solving this problem, and is provably optimal in cases where nothing is known *a prori* about the structure of the network and the signal routing method obeys certain reasonable constraints. The algorithm is efficient in practice, and is found to scale to networks of millions of nodes.

The difficultly of network reconstruction from coocurrences is analysed for graphs generated from a variety of random graph models, as well as a number of real world networks. Network structure is found to effect the reconstruction error rate, with scale-free networks resulting in the lowest error.

The problem of tree structured coocurrences is also considered. This is the case where the set of nodes in a coocurrence sample form a tree rather than a path in the network. We are not aware of previously published literature on this problem. A restricted form of the problem is proposed, and an algorithm is proposed to solve it. This algorithm can be considered a baseline for other work in this area, as it is simple and computationally efficient. Reconstruction using tree structured coocurrences is found to be harder than for path coocurrences, with higher error rates occurring.

Contents

Chapter 1

Introduction

Much of mathematics is concerned with the understanding of structure. By modelling the structure of physical phenomena we are able to reason abstractly; prediction and simulation becomes possible, and insight into the deeper nature of things follows. The study of networks, using the tools of graph theory, has become an essential modelling tool.

Many large systems may be modelled as networks. Chaotic natural processes, organic growth, systems whose structure differs on a micro and macro level can all be treated as networks. Networks have become a powerful tool for the study of semi-structured systems.

Many of the systems of interest are so large that exact determination of the network structure is impractical. This work is concerned with determining the structure of networks based on samples extracted either through passive observation or active probing of a network. The measurements are often indirect, the true structure can not be observed directly. Tasks of this kind are called *Network tomography* (Vardi [1996]), due to the conceptual similarity to reconstruction methods used in other fields to recover models by measurement in slices, such as medical imaging. As these methods deal with networks in the abstract, they have wide applicability. Most of the research has been conducted in the setting of computer networks, where modelling the topology of datacenters and the internet can allow for more efficient routing. However applications in genetics and in the modelling of social networks have been noted (Castro et al. [2004]).

As mentioned, it is the impracticality of making exact structure measurements that makes network tomography necessary. The most common type of measurement is done by propagating a signal through the network. The flow of this signal reveals something about the structure. Often we are able to determine the order that the signal traverses the network nodes, but with a level of imprecision. Perhaps the signal travels so fast that it appears instantaneous, or the signal follows multiple paths (or a tree) at once, and the timing information alone doesn't reveal the full structure. In this work we discuss both cases in depth, and we show some concrete applications of each.

Chapter 2

The Problem

2.1 Internal & External measurements

There are two main types of network tomography covered in the literature, that of internally sensed & externally sensed. The majority of research has been on the external case. This occurs when you have one source of signals, and wish to route signals through a network of unknown topology to a number of destinations out side that network. This is illustrated in Figure 2.1. Typically you assume a tree topology, and have no *a prori* additional information about the network's structure.

Internally sensed network tomography occurs when you have a number of sources and destinations as part of a larger network. The signal propagation between sources and destinations may be unicast or multicast. Typically it is possible to measure which nodes or links in the network a signal travels through. This situation is less common in computer networks, but may occur in telephone, biological or social networks. The

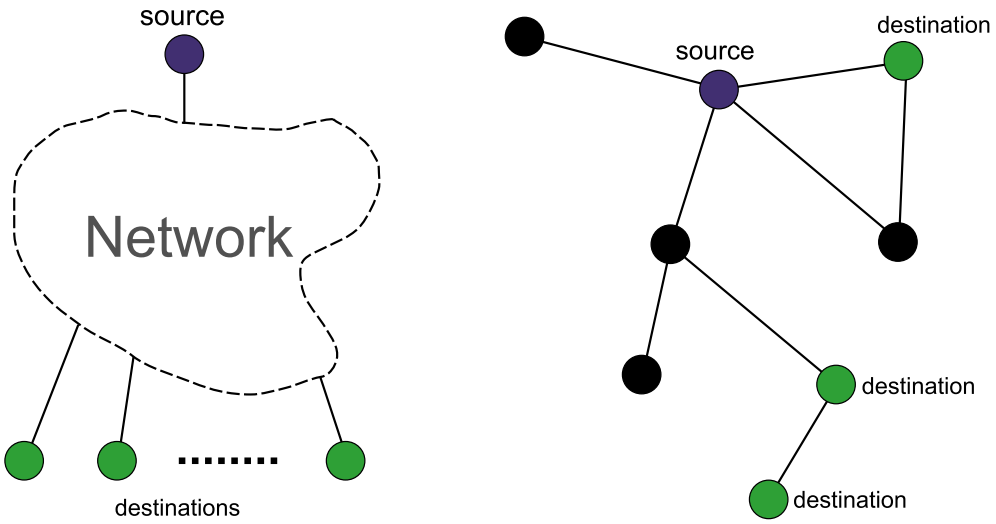


Figure 2.1: Externally and internally sensed networks.

problem is nontrivial when the order that the signal travels through the network is unknown.

2.2 Logical network topologies

It is important to make the distinction between physical and logical network topologies. The *physical topology* is the actual network topology, including network elements that are transparent to the signal flow. The elements in the logical topology are those that effect the path of signals in a detectable way. In the network tomography problems most often discussed in the literature and in this work, only the logical topology can be determined. The precise definition is different for internally and externally sensed networks.

For externally sensed networks, the logical topology consists of only those network elements for which signal paths can branch at (Vardi [1996]). No indication as to the existence of nodes not in the logical topology is given, so we can only hope to reconstruct the logical topology (see Figure 2.2). In the internally sensed case, sensors are placed on nodes (or equivalently edges) in the physical topology, and so only those with sensors are part of the logical topology. So in the internally sensed case branching need not occur at nodes in the logical topology.

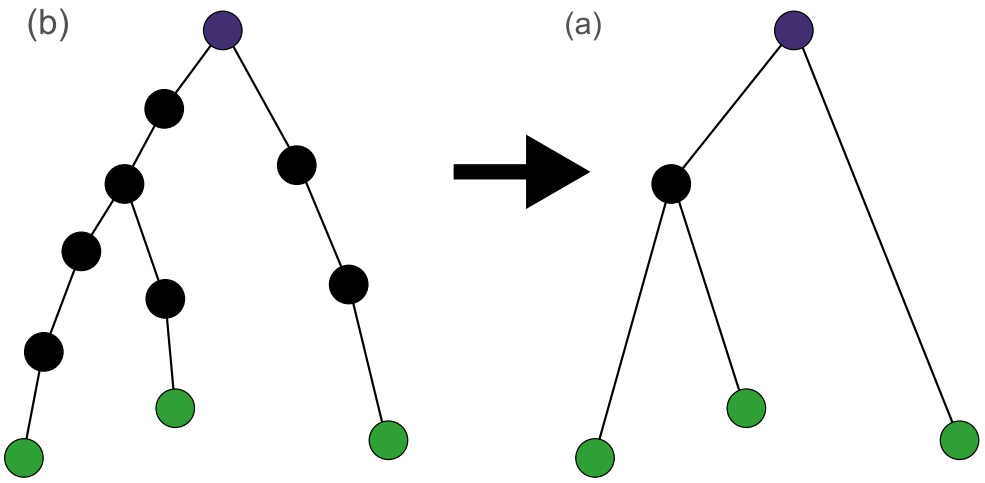


Figure 2.2: A physical network (b) and its corresponding logical network topology (a)

2.3 Network Inference using path coocurrence samples

Consider a graph, $G = (V, E)$ consisting of n nodes, along with a routing scheme R. A path coocurrence sample X_i on G is a subset of V, where there exists some path directly connecting the nodes in X_i, and that path was routed using scheme R. A routing scheme may be any algorithm for generating paths, or a set of constraints that each path must obey. We are not given any information about the ordering of the nodes. The network topology tomography problem is reconstructing as much of the topology of G as possible, given a set of path coocurrence samples.

In typical problems, the source & destination of each coocurrence path is known in the form of additional side information. The problem can also be framed without known destinations (just known sources). The known source and destination case is the easiest to solve due to the additional information.

Suppose we have k coocurrence samples (X_i, $i = 1..k$) which we represent as the indicator matrix X, where $X : k \times n$, The source and destination for coocurrence i are s_i and d_i respectively (for convenience we will not consider them as being in X_i, although they are part of the coocurrence).

When illustrating coocurrences, we use the compact notation "s-(a,b,...)-d" to represent a coocurrence with known source s, destination d, and 'inner' nodes a,b,... . So for example the coocurrence 3-(2)-5 has both a known source & destination, and since it only has one inner node, the order of the coocurrence's signal must have been 3, 2, 5.

2.3.1 Indistinguishable Node Sets

Before detailing methods for graph reconstruction from coocurrences, it should be noted in which cases exact reconstruction is impossible. A simple example is a single coocurrence sample with 4 nodes, including a source, destination and 2 inner path nodes. Without any other information, it is not possible to determine in which order the two inner nodes occur in the underlying graph (Figure 2.3). The terminology "indistinguishable" will be used to describe a set of nodes with this property. This differs from the case where nodes in the physical topology are hidden in the logical topology, as it is just the case that our coocurrences don't allow the nodes to be distinguished; they may be distinguishable given additional samples.

Figure 2.3: With only a coocurrence sample with source and destination as depicted, the inner two nodes would not be distinguishable

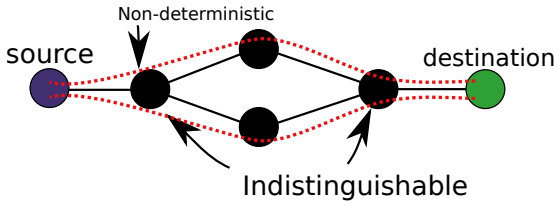


Figure 2.4: A pair of indistinguishable nodes in a non-deterministically routed network. Two possible signals are represented in red. This could correspond to the behaviour of a load balancing system. Merging these nodes may be undesirable as the reconstructed graph topology could then never be exactly correct.

One simple case of indistinguishable nodes can be determined easily during preprocessing of the samples. If two nodes x_i and x_j only ever appear in the same samples, then they form part of an indistinguishable set. Given this easy test, it might seem reasonable to "merge" or otherwise consider each set of indistinguishable nodes as one. This merging corresponds to the graph-theoretic notion of vertex contraction. However, care must be taken. In the case where the underlying coocurrence paths are generated using deterministic, shortest path routing, the nodes of a coocurrence set form a subpath in the graph, and merging them together is sensible. However, when non-deterministic routing is used the nodes may be very far apart, and by merging the nodes together, the correct logical network topology is no longer a possible reconstruction. This is illustrated in Figure 2.4.

Unless otherwise stated we assume that deterministic routing is used for coocurrences, and that before a reconstruction algorithm is run each indistinguishable node set is merged together, with the set taking on the index of the lowest indexed member. So two or more nodes are replaced by a singled node, whose neighbours are the union of the neighbours of all the nodes in the merged set. Without this step, the reconstruction has a higher error, and the knowledge that the sets of nodes are indistinguishable is useful information, so in practice one would always want to run such a preprocessing step.

It is important to note that the above case where a set of nodes always appear together is not the only case where nodes are indistinguishable. Another uncommon case is when in all but one sample a pair of two nodes appear together. If in the extra sample all the other nodes don't appear in the other coocurrence, then they give no additional information about the ordering of the node pair. So they are still indistinguishable to any reconstruction. Examination of a large number of coocurrence samples has lead the author to believe that such cases are the main cause of reconstruction errors in the algorithms described later in this chapter. See Figure 2.5 for an example.

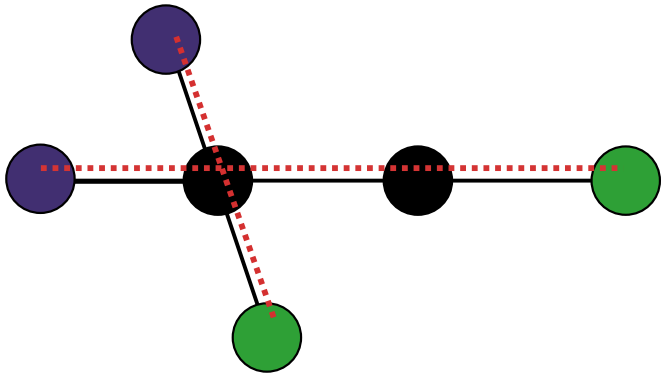


Figure 2.5: The two black nodes shown do not all ways occur in the same coocurrences (dashed red lines). In the case of the horizontal coocurrence, the order in which the signal goes through the two black nodes can't be determined from the two coocurrences.

2.4 Frequency weighting method

This method, originating in Rabbat et al. [2005], is based on the intuitive principle that if a node often appears in a coocurrence involving a source s, then it is likely close to that source in the underlying network. A similar argument holds for destinations. This might seem like a loose assumption, but consider the case when the signals that create the coocurrence are routed using a shortest path scheme. Let a and b be two nodes near a source s, when the shortest path from s to b goes through a. Then any coocurrence involving s and b must also contain a and so weighting each node's distance from s based off of the number of times they occur together makes sense. a will always be weighted as closer or equal distance from s than b. For this reason, the weighting performs quite well. This method will be used as a baseline for comparisons against other methods.

The algorithm is as follows. For each coocurrence j, each node m in that coocurrence is assigned weight $\alpha_j(m)$. Following the intuition above, a higher weight will imply that m is nearer j's source. $\alpha_j(m)$ is calculated as

$$\alpha_j(m) = x_{s_j}^T x_m - x_{d_j}^T x_m$$

The notation x_m means the mth column of the matrix X, which can be though of as an indicator vector, where $(x_m)_j = 1$ implies that the mth node is part of the jth coocurrence. Essentially $\alpha_j(m)$ is just the number of times m occurs with s_j as the source minus the number of times m occurs with d_j as the destination.

Once we have the weights α we reconstruct the graph by assuming that the signal that generated each coocurrence travelled through nodes in order of decreasing α. Note that as before we don't include the source and destination as part of the coocurrence, so no α value is computed for them.

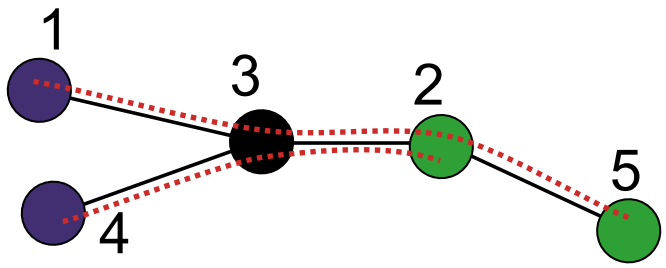


Figure 2.6: Graph with two coocurrences, 1-(2,3)-5 and 4-(3)-2

When two nodes in a coocurrence have the same α value, we cannot determine the order in which they occur using this method. This may be because the nodes are indistinguishable (See 2.3.1), or just a failure of this method to distinguish them. A random choice of the order of the two (or more) nodes is reasonable when implementing this method.

It might seem at first to be an optimal reconstruction strategy for shortest path routing, but in many cases it is possible to do better, as this method draws no information from overlapping coocurrence that don't share a source or destination. An example of this flaw is illustrated in Figure 2.6. The coocurrence 4-(3)-2 tells us that there must be a link between nodes 3 and 2. For the coocurrence 1-(2,3)-5, the FM algorithm gives equal weight to 2 and 3, and so cannot determine their order. A better algorithm would make use of the information that an edge exists between 3 and 2 from the other coocurrence, and would reuse that edge when reconstructing the coocurrence 1-(2,3)-5. The reason why such a reconstruction is better than the alternative is discussed in Chapter 3.

While this algorithm can be adapted to give a set of possible reconstructions, corresponding to all possible tie braking choices for when α values are the same, the resulting sets may not contain the actual network structure. Consider the graph & coocurrences in Figure 2.4. In this case the α value for the two indistinguishable nodes will be greater than the pair of load balanced nodes in between them, so both indistinguishable nodes will be placed nearer the source. This can occur with distinguishable nodes as well, so grouping or merging indistinguishable nodes doesn't remove this problem. When deterministic shortest path routing is used, it is easy to see that nodes with the same α value within a coocurrence must form a subpath, so at least the correct solution is one of the possibilities.

2.5 Maximum likelihood approach

The frequency weighting method above could be considered a local greedy method. It uses global information about the weights, but reconstructs each path separately. By treating the samples as probabilistic, a global optimisation approach becomes more natural. This section describes the state of the art method for topology reconstruction

from coocurrences, detailed in Rabbat et al. [2006], which takes such an approach.

Coocurrence paths can be thought of as random walks on the underlying network. A signal starts at some source, then transitions to an adjacent node, with each edge out of the node considered as weighted with the probability of transition through that edge. The signal continues through the network until it ends at some node, which is called the destination. This method does not model signal attenuation; the length of the path does not directly effect that path's probability of being generated. As we know from the size of the coocurrence samples what the length of the paths are, we don't need to model them. This model of probabilistic transitions corresponds to a first order Markov chain, and indeed the weighted incidence matrix gives the parameters of the Markov chain. In this document we are considering the case of known source and destination, so the initial state probabilities do not need to be modelled.

Random walks might seem a poor model for highly structured paths such as those in computer networks. As paths are not true random walks, the transition probabilities are better interpreted as just edge statistics. If the coocurrence samples are representative of actual traffic through the network then this is reasonable (such as for passively collected samples). This method also has the advantage of producing sparse, low degree reconstructions. Edges not necessary for the reconstruction will not appear, as they would only reduce the path probabilities, leading to a less optimal solution. So no regularising term is needed to prevent over-fitting.

When the transition (incidence) matrix is known, we can reconstruct each path by taking the most likely ordering. If two orderings are equally likely, a random one of the two can be taken. The number of orderings scales with the factorial of the path length, so some care must be taken for long paths. A dynamic programming algorithm can be used to reduce this calculation to polynomial time complexity.

Figure 2.7 shows a pair of coocurrence samples and the transition matrices for both. As the second coocurrence sample makes use of the edge $3 \to 2$, and intuitively we would expect this edge to be more likely than the edge $2 \to 3$ in the first coocurrence, all else being equal. Under the random walk model the path probabilities both come out as 1 when we assume the graph has that structure (as in the graph in Figure 2.8), whereas for the other possible structure the path probabilities are both 0.5. In this case we find that the maximum likelihood gives the (correct) intuitive reconstruction.

2.5.1 Problem formulation

As before, let the matrix of coocurrence samples be denoted X. The number of nodes in each coocurrence is then $N_i = \sum_{j=1}^{n} X_{ij}$. The true (unknown) ordered path for each coocurrence is $z^{(i)} = (z_1, z_2, \ldots, z_{N_i})$, the ordering of the nodes in each path by index is $y^{(i)} = (y_1, y_2, \ldots, y_{N_i})$, and the corresponding permutation between the two is $\tau^{(i)} = (\tau_1, \tau_2, \ldots, \tau_{N_i})$, such that $z_t = y_{\tau_t}$. The paths y can be considered the default ordering, and the reconstruction task becomes finding some permutation τ of y that gives the highest likelihood.

1-(3,2)-5
4-3-2

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.7: Example transition matrices for a coocurrence problem

As stated before, for this method we model the paths as random walks with a (binary) transition matrix $A : n \times n$. The total transition probabilities for each node must sum to one, so we have the additional constraint that

$$\sum_{j=1}^{n} A_{ij} = 1, \quad \text{for each node } i$$

Assuming that the permutation of each coocurrence sample is equiprobable and independent of the path, we find that the likelihood of a particular coocurrence path y given a permutation τ is

$$P[y \mid \tau, A] = \prod_{t=2}^{N} A_{y_{\tau t-1}, y_{\tau_t}}$$

or

$$P[y \mid \tau, A] = \prod_{t=2}^{N} A_{z_{t-1}, z_t}$$

Both are equivalent, the first makes the dependence on τ clearer. We want the likelihood in terms of A only, so we need to marginalise out τ (Summing over all possible permutations of N elements, which we denote Ψ_N)

$$P[y \mid A] = \frac{1}{N!} \sum_{\tau \in \Psi_N} P[y \mid \tau, A]$$

So under the assumption that each coocurrence sample is independent, the complete

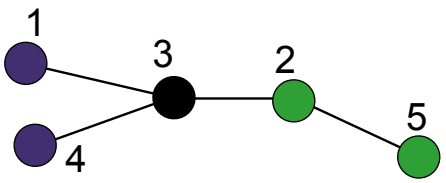


Figure 2.8: Correct reconstruction for the coocurrence problem in Figure 2.7

likelihood is

$$P[y \mid A] = \prod_{i=1}^{k} P[y^{(i)} \mid A]$$

and the log likelihood is then

$$\log P[y \mid A] = \sum_{i=1}^{k} \left[\log \left(\sum_{\tau \in \Psi_{N_i}} P[y^{(i)} \mid \tau^{(i)}, A] \right) - \log(N_i!) \right]$$

Several interesting things can be noted about this expression. Firstly note that the inner sum is over all permutations of N elements. As this grows with the factorial of N, just evaluating this function for a particular A and y is slow for long paths. Secondly, note the sum within the inner logarithm. This prevents the function from being simplified further. The EM algorithm (Bishop [2006]) is the standard method for solving problems where this occurs, and it is applicable here, where τ is treated as the hidden variable.

There are several downsides to using the EM algorithm here. In general it only returns a local optimum of the likelihood function, so the algorithm must be run multiple times with different starting values, with the best optimum found being kept. The EM algorithm doesn't avoid the need to sum over all permutations, and in fact this is required at the E step during each iteration.

In Rabbat et al. [2006], the exact formulation of the E and M steps are given, so they won't be detailed here. They also give an approximate E step, which avoids the sum over all permutations. In this document we will only perform comparisons against this method using the exact E step, to ensure best performance. Unfortunately we cannot run this method on large problem instances using the exact E step.

Chapter 3

An Integer Programming Solution

The solution space for the network reconstruction problem discussed is finite. Given enough time, the full set of possible reconstructions consistent with the data can be enumerated. If such an enumeration was feasible, the question naturally arises: What test can we apply to determine which solution is the optimal solution? No such property of a reconstructed graph is immediately obvious. The ML method (Section 2.5) attempts to maximise the likelihood of the paths under a probabilistic walk model. While more effective than the FM method's path local reconstruction (Rabbat et al. [2006]), it still makes little use of the structure of problem.

The principle of Occam's razor could help us here. Occam suggested that when given multiple solutions consistent with the data, all else being equal, we should take the simplest. As any permutation of the coocurrences give as a reconstruction consistent with the data, we can apply this principle here. Defining the simplest graph however, is difficult. If we have some prior knowledge about probable network properties, for example clustering or node degree distributions, then these could perhaps be used. For the purposes of this section, we don't assume any *a prori* knowledge. Without knowledge of the graph's possible structure, the obvious measure of the graphs complexity then becomes the edge count. A graph with fewer edges is 'simpler'.

In this chapter we will show a number of remarkable properties of the simplest reconstruction. Under some mild assumptions about the coocurrence's signals, the correct graph reconstruction has the minimum number of edges out of all possible reconstructions. However, there may be many possible reconstructions that obtain this minimum, and we must naturally ask if it is possible to further distinguish between them. It turns out that any of the minimum edge reconstructions corresponds to equally likely underlying graph, so without additional assumptions, they cannot be distinguished.

Given these results, the problem then becomes finding the graph with the least number of edges consistent with the coocurrence data. In Section 3.2 we give a set of linear constraints that encodes this consistency, so that any graph that obeys the constraints is a possible reconstruction. Under some mild assumptions about the signals that generated the coocurrences, it will be shown that these linear constraints encode all

possible reconstructions as well. Finding the minimum edge graph then becomes a binary integer programming problem.

In the next Chapter, we show that this BIP method is often solved by its LP relaxation, and that the overhead of using a branch and bound LP method over just the LP relaxation is negligible. Empirical comparisons against the two methods described in the previous chapter show that the BIP method performs better, suggesting the neither of those two methods returned the optimal reconstruction.

A method is also proposed for finding all best reconstructions under the above mentioned assumptions. An alternative version of the BIP method is also considered that can handle graphs that don't obey these assumptions.

3.1 Minimum edge reconstruction

Definition 1. *A coocurrence/path on an undirected graph is* ***shortcut free*** *if the subgraph induced by the coocurrence/path is acyclic. A graph reconstructed from a set a coocurrences is shortcut free if the coocurrences are shortcut free when considered as originating on the reconstructed graph. The shortcut free routing scheme just specifies that all routed paths must be shortcut free.*

This shortcut free property is important to the method that follows. By *induced subgraph* we mean the set of nodes from the coocurrence and the set of all edges in the graph between pairs of nodes in the coocurrence. So informally what this definition says is that a coocurrence is shortcut free if there is no edge in the underlying graph

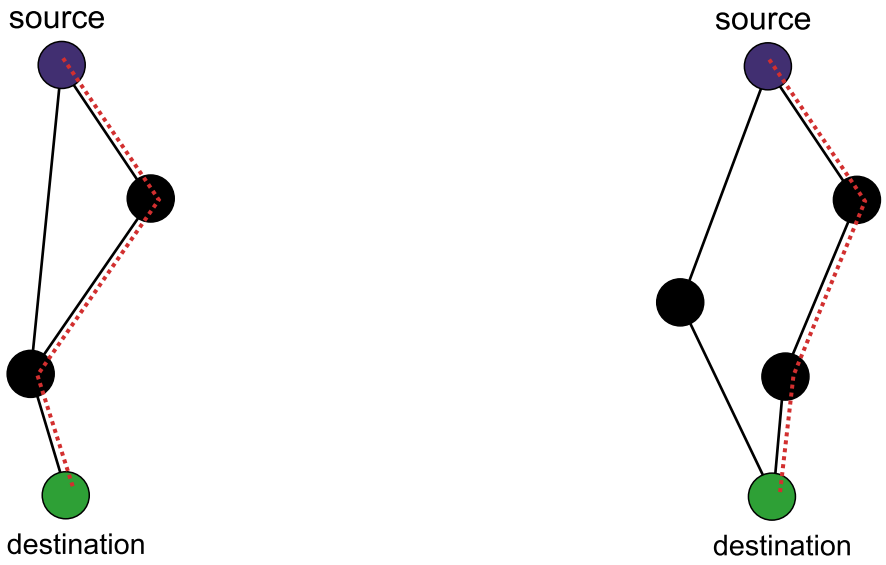


Figure 3.1: An example of a coocurrence shortcut on the left, and a coocurrence that is not routed using unweighted shortest path routing, but is still shortcut free on the right

that skips nodes of the coocurrence. This is essentially a weak statement about the quality of the signal routing. Only a badly routed signal would go through unnecessary nodes in such a way. It is important to note that this is much weaker than an assumption of shortest path routing on an unweighted graph. Figure 3.1 illustrates the difference. There could still be a shorter path between the source and destination of the coocurrence, it just must make use of at least one node that is not in the coocurrence. The shortcut free property could also be though of as a more local version of the (unweighted) shortest path assumption.

Note that on a graph embedded in Euclidean space whose edge weights are induced by a metric, shortest path routing implies paths are shortcut-free, due to the triangle inequality. For arbitrary edge weights, this is only the case where routing is deterministic, in the sense that when there are several paths between two nodes of equal length, the same path is always taken. We call a coocurrence a *shortest path* coocurrence if there is no shorter path between the coocurrence's source and destination in the underlying graph.

Definition 2. *Given a graph $G = (V, E)$ and a set of coocurrences, the **Clairvoyant reconstruction** is the subgraph $G = (V, E')$ consisting of the all edges in paths that generated the coocurrences.*

The clairvoyant reconstruction is the best we can hope to do. It contains edges that are in some way witnessed by the coocurrences. The full graph may contain edges that are never used in a coocurrence path, and so we cannot hope to reconstruct those edges. We use the term *coverage* to denote the percentage of edges that are in the clairvoyant reconstruction compared to the logical network.

Theorem 1. *For a set of coocurrence samples $X_1 \dots X_k$ whose routing is shortcut free and deterministic, no consistent reconstruction can have fewer edges than the clairvoyant reconstruction.*

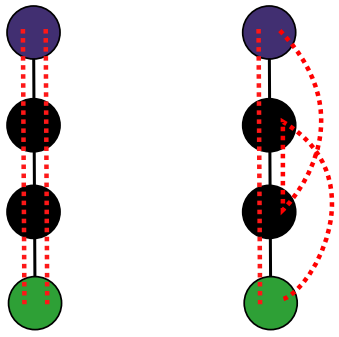


Figure 3.2: An example of how a reconstruction of a pair of coocurrences will have shortcuts if it does not share as many edges as possible between the two coocurrence's reconstructed paths. Notice that the coocurrence paths on the right both have 2 shortcuts

Proof. As the length of each coocurrence's path is fixed by the size of that coocurrence, the only way for one reconstruction to have fewer edges than another is if more edges are reused between pairs of coocurrence paths. We will show that the clairvoyant reconstruction reuses all possible edges between paths that can be reused, so that no other consistent reconstruction can have fewer edges.

Let X_i and X_j be any two of the coocurrences. The minimum edge reconstruction for the pair will reconstruct the edges between the set of nodes $X_i \cap X_j$ in the same way for both coocurrences. Any reconstruction that does not will have a shortcut in the induced subgraph for X_i or X_j or both. To see why, just notice that the subgraph induced by X_i will have more than $(n-1)$ edges, due to the extra edges used in X_j. This is illustrated in Figure 3.2. So as the coocurrences are shortcut free, the clairvoyant reconstruction must also have the same number of shared edges between the two coocurrences as the actual network.

So we have that the clairvoyant reconstruction has as few edges as is possible for each pairwise sub-reconstruction. A reconstruction that achieves the minimum edges for every pair will give a lower bound on a global minimum.

□

Theorem 1 above gives a lot of insight into the nature of coocurrence reconstruction. Without some assumptions about the structure of the underlying network or the path routing, there is no way to distinguish between the each of the possible consistent reconstructions. When we intuitively try to reconstruct these coocurrence paths, we try to reuse edges that are 'witnessed' by other coocurrences. Theorem 1 tells us that under the shortcut free assumption, this is a reasonable thing to do.

Following a similar argument to that in Theorem 1, note that a minimum edge reconstruction is shortcut-free, and so given no further assumptions about the network structure, its not possible to distinguish between a set of minimum edge reconstructions.

3.2 Encoding feasible reconstructions with linear constraints

In the previous section we showed that the shortcut free assumption seems to encode our intuition about good reconstructions. The question then becomes how to encode this assumption as a set of constraints, so that we may use a optimisation algorithm to find the minimum edge graph that is consistent with the data & shortcut free. In this section we show that this can in fact be encoded in a set of linear constraints in the case of an undirected graph.

The constraints are as follows, for each coocurrence:

Degree constraint Within the subgraph induced by the coocurrence, the source and destination have degree 1, and the inner nodes have degree 2.

Connectivity constraint For any partition of the coocurrence into two sets of nodes, there exists an edge between the two sets.

Theorem 2. *The connectivity and degree constraints for a set of coocurrences are satisfied by a undirected graph if and only if it is consistent with the coocurrences, and shortcut free.*

Proof. Let $X_1 \dots X_k$ be a set of coocurrence samples. Let G be a reconstruction consistent with the coocurrences and shortcut free. Consider each coocurrence X_i. As the coocurrence is connected in G (this is implied by it being consistent) it must satisfy the connectivity constraint. Similarly the degree constraint is satisfied, as the induced subgraph is a path, and paths satisfy the required nodes degrees.

Let G be some graph that obeys the connectivity and degree constraints for the given set of coocurrences. Consider each coocurrence X_i. Let G_i be the subgraph of G induced by X_i. Then G_i is connected, as any disconnected graph can be partitioned into two sets of nodes such that there is no edge between the two sets, which would violate the connectivity constraint. So as G_i is connected, it is easy that the degree constraints imply that it is a path. So X_i is shortcut free, and consistent with G. □

These constraints are easy to encode as linear equalities in the undirected case. We first assume that the set of edges is given as a binary indicator vector x, which for clarity we index using the notation $x_{i \to j}$, which denotes the undirected edge between nodes i and j. Each coocurrence X_i is encoded separately as follows:

For coocurrences with 2 nodes—just the source and destination—the only constraint needed is $x_{s_i \to d_j} = 1$. For the case where the coocurrence has more than 2 nodes, the following is used:

Degree linear constraints

$$\sum_{v \in X_i} x_{s_i \to v} = 1$$

$$\forall v \in X_i, \sum_{u \in \{s_i, d_i\} \cup X_i} x_{v \to u} = 2$$

$$\sum_{v \in X_i} x_{d_i \to v} = 1$$

Connectivity linear constraints

$$\forall \text{ partitions } U, V \text{ of } \{s_i, d_i\} \cup X_i \colon \sum_{u \in U} \sum_{v \in V} x_{u \to v} \geq 1$$

The set of constraints for all coocurrences can easily be written with matrix notation as two statements $Ax = c$ and $Bx \geq d$, where the first encodes the degree constraints and the second the connectivity constraints. For the degree constraints, the number of rows of A need for each coocurrence is just $|X_i| + 2$. The connectivity constraints are a different matter unfortunately. The number of ways to partition a set of size n into

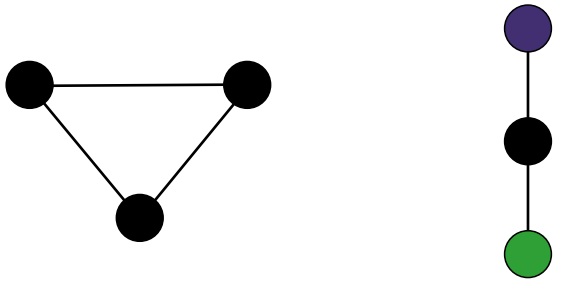


Figure 3.3: An example of how the degree constraint can be satisfied, but the resulting induced subgraph is not a path. Notice the two parts consist of a ring (on the left) and a short path.

Table 3.1: Scaling of number of connectivity constraints with the size of each coocurrence

Coocurrence size	4	5	6	7	8	9	10	11	12
Naive partitionings	7	15	31	63	127	255	511	1023	2047
Degree satisfying partitions	0	0	4	15	41	98	218	465	967

two parts is given by the Stirling number of the second kind (Knuth [1997]), $\{{n \atop 2}\} = 2^{n-1} - 1$, and so the number of rows of B needed for each coocurrence is exponential in the size of the coocurrence.

The number of connectivity partitions needed can be reduced by only including partitions that could satisfy the degree constraints. Consider a partition U, V. In the case where one of U or V contains both s_i and d_i, the other set must be a ring, and so must have at least 3 vertices. See Figure 3.3 for an example of a possibility. Similarly partitionings where s_i and d_i are in different partitions need not be considered. For the ring case, we can see that there are $\binom{X_i}{1} + \ldots + \binom{X_i}{X_i-3}$ possibilities, calculated as the number of ways to take vertices out of the full X_i sized ring and move them into the other set.

The requirement for an exponential number of constraints initially seems like a serious flaw. However, there are a number of factors that mitigate this problem:

- Most real world graphs exhibit some sort of small-world phenomenon (see Section 5.1). The graph diameter is typically on the order of $\log n$, and almost always less than 12. Given a known source and destination this would result in roughly 1000 constraints for a very long path, and usually much less (see Table 3.1). Simulations suggest that the number of connectivity constraints needed is practical even for large graphs (see Chapter 4).
- Most large coocurrences do not overlap completely with other coocurrences, and so many of the nodes are merged as described in Section 2.3.1, making the coocurrences effectively shorter.
- In practice, just using the degree constraint is often enough. During extensive testing we found that disconnected paths were very rarely returned by optimisation algorithms using just the degree constraint. This appears to be due to

overlap between coocurrences, as with the point above. This is detailed in Section 4.6.

- Suppose we only use the degree constraint. In cases where an optimisation algorithm returns a graph where one or more coocurrences are not connected, the connectivity constraints that were violated can be introduced and the algorithm re-run. This is a form of constraint generation (see section 3.7).

The structure of the constraints may seem familiar. They share properties with many existing optimisation problems. In particular, the degree constraints are reminiscent of the 2-matching problem, and the connectivity constraints are effectively encoding a statement about graph cuts.

A *k-matching* is a set of edges such that each node is incident to at most k edges in the set. A perfect matching is just a matching where each node is incident to exactly k edges. Matching problems are a well studied area, and there exists efficient algorithms to solve them. Unfortunately, the degree constraints are sufficiently different that these algorithms can't directly be applied. The main problem is that the constraints for the matching problem have to be in the form of a node-edge incidence matrix.

Definition 3. *The **node-edge incidence** matrix for a graph, is a* $|V| \times |E|$ *binary matrix* X *where* $X_{i,j} = 1$ *if node* i *is incident to edge* j*.*

Node-edge incidence matrices have the key property that there is either 0 or 2 ones in each column. In the degree constraints each row does correspond to a node, however there can be several constraints for a single node, each corresponding to a possible path (and pair of edges) through that node. This difference is fundamental. The clairvoyant graph could not have a perfect 2-matching, while still obeying the degree constraints.

The connectivity constraints consider partitions of the nodes of the coocurrence into two sets. This is essentially the same concept as a *graph cut*. The problem of finding the minimum cut of a weighted graph with sources and destinations (i.e., a flow network) is another well known algorithm for which polynomial time solutions exist. While our problem formulation also makes use of the idea of sources and destinations, it is still fundamentally different from the standard graph cut or max flow problem. For a flow network, with multiple sources and destinations (sinks), there is no concept of requiring a flow to come from a specific node, at each sink. For similar reasons the connectivity constraint is more complex than for the linear programming formulation of the minimum graph cut problem.

3.3 A binary integer programming formulation, and its relaxation

The constraints formulated in the previous section are well suited for use in an optimisation algorithm. The most straightforward of which would be a binary integer program. *Integer programming* is the solving of constrained optimisation problems with linear constraints and weights, where the variables are restricted to integer values. We are only concerned with binary valued variables (representing the edges). The binary integer programming formulation of the coocurrence problem using the linear constraints from the previous section is:

$$\min_x \mathbf{1}^T x \quad \text{subject to } Ax = b \text{ and } Bx \geq d$$

where x is the binary indicator variable for the edges, and the constraint matrices are of the form described in the previous section. The bold 1 represents the vector of all ones.

The general class of binary integer programming (*BIP*) problems is *NP*-hard, so solving the above BIP directly could be slow. Many classes of BIP problems can be solved efficiently by using the linear programming relaxation, which is formed by changing the constraint that each x_i be binary to $0 \leq x_i \leq 1$, which gives a linear programming problem. For the coocurrence problem, this gives

$$\min_x \mathbf{1}^T x \quad \text{subject to} \quad \begin{aligned} &Ax = b, \\ &Bx \geq d, \\ &0 \leq x_i \leq 1 \quad \forall x_i \end{aligned}$$

For some classes of problems this gives an exact solution, as a linear programming solution gives 0 or 1 for each x_i (Nemhauser and Wolsey [1988], Chapter II.3). On other problems this gives a solution that can be rounded to within a certain factor of the BIP problem's minimum (Papadimitriou and Steiglitz [1998], Chapter 17). Note that this depends on the use of Simplex or interior point methods, for reasons which are described later. There is a large and diverse body of research into integer programming and linear relaxations, as many of the combinatorial problems that are of great importance in industry (scheduling, traveling salesmen problem, etc.) can be formulated naturally as binary integer programs (Nemhauser and Wolsey [1988], Chapter 1).

The question that naturally arises given the BIP formulation is how effective is the linear programming (*LP*) relaxation for solving it. In simulations described in detail in Section 4.6, it seems that the linear program almost always returns binary values for the variables. This is an interesting result which is not at all obvious from examination of the BIP. The similarities to common linear programming problems noted above gives an indication why the LP relaxation is often effective. We also note in Section 4.6 that when using a branch and bound based solver which uses LP relaxation as an intermediate step, the running time is only 5-10% higher than using just the LP

relaxation, when tested on those problems for which the pure LP relaxation often fails. On small-world graphs more than 99% of the time the LP relaxation gives integer solutions (see Section 4.6).

To attempt to explain why the LP relaxation is often effective for this problem, we need to discuss conditions under which IP problems are exactly solved by there relaxations. The most common test is to determine if the *polytope* described by the set of constraints is *integral*:

Definition 4. *A **polytope** is a geometric object with flat sides. In the context of linear programming, they are convex, exist in the solution space of the problem, and are described by a set of linear equalities as* $\{x \mid Ax \leq b\}$*, for some* A *and* b*. A polytope is **integral** if every vertex is described by integer coordinates.*

The requirement of a integral polytope is a strong condition. For the LP relaxation to exactly solve the BIP it is sufficient for only the solution vertex to be integral. For constraint matrices with values in $-1, 0, 1$ the condition of *total unimodularity* implies that the polytope is integral, see Papadimitriou and Steiglitz [1998] for details. In the case of binary matrices the more general condition that the constraint matrix be *balanced* can be used:

Definition 5. *A binary matrix is* A *is **balanced** if **no** submatrix* A' *satisfies the following conditions:*

- A' *is* $m \times m$ *where* m *is odd.*
- *Each row and column of* A' *sums to 2. I.e.* A' *is the incidence matrix of graph cycle.*

For any balanced matrix A the polytope $\{x \mid Ax \leq b\}$ is integral for any b (see Papadimitriou and Steiglitz [1998] p571 for a proof). For the coocurrence problem's LP formulation, the constraint matrix is unfortunately not always balanced. If any consistent reconstruction's graph contains a cycle of odd length, then a submatrix of the form described in Definition 5 might exist. This occurs in cases where the inner parts of the coocurrences overlap, see Figure 3.4. It is easy to see that the node-edge incidence matrices of bipartite graphs are balanced, so at the very least the coocurrence LP solves the BIP problem if all consistent reconstructions are bipartite.

The reason why the odd order, 2-cycle submatrices in Definition 5 are a problem is simple. Solving an equation of the form $A'x = b$ will give fractional values for the variables in x for some values of B.

3.4 Obtaining all optimal reconstructions

> If several theories are consistent with the observed data, retain them all.
>
> *Epicurus (341 BCE - 270BCE)*

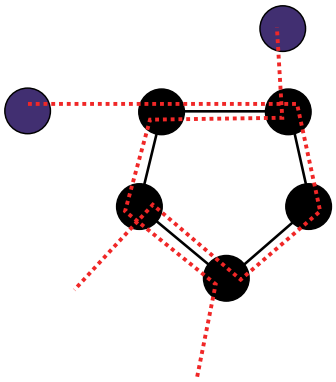
Using any one of the three methods discussed, a solution may be found that is consistent with the coocurrence data. For real world application of these methods, some guarantee as to the quality of the solution is desirable. One useful measure would be the number of solutions that are equally good, in some sense, as the returned solution. This would give an idea as to the uncertainty in the underlying coocurrence data, as a large number of good reconstructions suggests that the coocurrences do not sufficiently define the graph. Even better would be to return the full set of good solutions, in line with the principle of Epicurus (see Hutter [2005]), which states that when there are multiple models consistent with the data, all of them should be kept.

This is possible with the FM method. When a set of weights for a particular path have nodes with the same weight, the method cannot distinguish in which order they should go. Returning all orderings, or the count of them, provides useful additional information. However, due to the simplicity of the method, the set of returned solutions is larger than the real optimal set.

The maximum likelihood method unfortunately has no clear way of returning all good solutions. Due to the highly non-convex search space the best we can do is return solutions that are found during the multiple restarts, and perhaps other points on the same plateau as the optimal found.

The BIP method introduced in this work does not run into the same problem as the EM method. As the constraint polytope is convex, it becomes possible to find all optimal (in the sense of minimum edge) solutions. The main caveat is that we are still restricting the search to shortcut free reconstructions.

Methods for returning the full set of optimal solutions to integer programming problems are not as widely used as one might assume. They are not a part of most of the standard numerical computing software packages. As our method is a binary programming problem, we can use the simplest approach, that of using *binary cuts*. The basic idea is start with a feasible solution to the particular BIP problem, then to add additional constraint/s so that the given solution becomes infeasible, while still main-



$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Figure 3.4: The left diagram shows a portion of a graph containing an odd length cycle. The matrix to the right is a submatrix of the degree 2 constraints. This shows that the full constraint matrix is not balanced.

taining the other possible optimal solutions as feasible. The problem is then solved again, to get another feasible solution. This procedure can be done repeatedly, until the solver returns a suboptimal solution.

The particular constraint we used is due to Balas and Jeroslow [1972]. Suppose the initial optimal solution is the set of edges θ. Then to make θ infeasible, we add the constraint:

$$\sum_{(i,j)\in\theta} x_{i\to j} - \sum_{(i,j)\notin\theta} x_{i\to j} \leq |\theta| - 1$$

It is easy to see that θ violates this constraint, as the left hand side sums to $|\theta|$. Any other optimal solution must either have additional edges over θ, in which case the subtractive term makes the inequality true, or some edges in θ are not in it, in which case the sum over θ will be less than $|\theta| - 1$.

In most BIP applications solving the problem with the addition of this constraint is fast. If a linear programming based method such as *branch and cut* is used, the dual solution to the LP can be utilised. One of the results from the theory of duality for linear programs is that linear programs obey weak duality, in the sense that the value of any solution to the dual problem is a bound on the solution to the primal problem (Papadimitriou and Steiglitz [1998]). The upside of this is that the old optimal solution from before the cut was added can now be converted to a feasible point for the dual problem. This solution is in practice very close to the new optimal solution, and so it takes very few steps of the Simplex algorithm from that starting point. We won't discuss this further in this work, as these sorts of dual techniques are now standard (see Nemhauser and Wolsey [1988]).

As discussed in Section 3.3, the linear programming relaxation of our BIP almost always gives an integer solution. Unfortunately, after adding the binary cut this may no longer be true. In our numerical simulations, we found that a branch and cut method for solving the resulting BIPs performed well, taking almost no additional computational time than our LP solver. It tended to require only a few branches each time, often none. So this method is computational feasible, and even fast.

The number of possible optimal reconstructions is (in pathological cases) exponential in the path lengths. One such case is when all the coocurrences are disjoint, as then all possible orderings are equally good. In practice this is not a problem. In Section 5 we show how the distribution of the number of solutions varies with graph properties. The left histogram in Figure 3.5 gives a taste of the typical distribution for a typical class of graphs. Merging indistinguishable nodes during preprocessing of the problem reduces the running time further, while still allowing all possible optimal solutions to be recovered. Figure 3.5 illustrates how it reduces the number of iterations of binary cutting method required by a substantial amount.

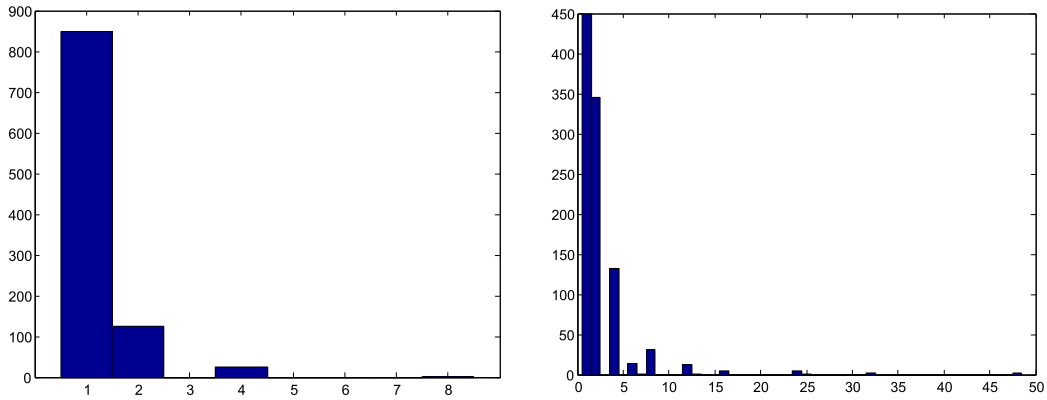


Figure 3.5: These two histograms show the distribution for the number of solutions over 1000 randomly generated graphs. For the left graph, the coocurrences were preprocessed to merge indistinguishable node sets, and for the right graph they were not.

3.5 Non shortcut free reconstruction with the *weak* BIP method

So far the BIP method described assumes that the coocurrences obey the shortcut free property (Section 3.1). While this property provides an interesting middle ground between chaotic routing and perfect shortest path routing, it can't be expected to hold in all cases. The constraints that the BIP method uses will only hold with equality for reconstructions with the shortcut free property, and so the BIP reconstruction will provide suboptimal results or fail altogether when the shortcut free property is violated. In this section we discuss the case where the constraints are weakened further, so that such cases can be handled.

Suppose that the degree constraints in the BIP method are weakened to inequality constraints, in the following way:

$$\sum_{v \in X_i} x_{s_i \to v} \geq 1$$

$$\forall v \in X_i, \sum_{u \in \{s_i, d_i\} \cup X_i} x_{v \to u} \geq 2$$

$$\sum_{v \in X_i} x_{d_i \to v} \geq 1$$

We call this the weakened degree constraints, and the BIP reconstruction with these constraints the weak BIP method. Note that we are still considering the undirected edge case.

The first thing to note about the weak BIP constraints is that the reconstructed paths are not necessarily consistent with the coocurrences, in the sense that the set of nodes in a coocurrence may not form a path in the graph. See Figure 3.6 for an example. However, the constraints are still strong enough to the remove the possibility of trees whose leaves are not the source and destination nodes, as such nodes would violate

the middle constraint, with right hand side summing to 1.

Algorithms using the weak degree constraint can not use the smaller set of degree satisfying constraints previously discussed, as this is not sufficient to ensure connectivity. The full set of constraints can still be avoided, as it is easy to see that partitionings with 1 node in a partition, or 2 nodes (excluding the case of {source, destination}) do not satisfy the weak degree constraints. Cases where the source and destination are in different partitions are now possible though.

When we discussed the connectivity constraints previously, we noted that reconstructions violating connectivity when only the degree constraint was used were exceedingly rare. It seems that a similar phenomena holds for reconstructions using the weak degree constraint that violate path consistency. Experimental evidence in Chapter 4 show that the error rate with the weak BIP method is not much higher than with the strong BIP method.

3.6 Optimisations & efficient implementation

An exact implementation of the method as described is reasonably efficient. The main implementation details that hamper performance are large vectors and matrices. While using a sparse matrix and vector representation should in theory elevate these problems, in practice linear programming solvers and associated tools perform faster when feed smaller matrices. Section 4.3 shows how using sparse but unoptimised code performs more than 70 times slower for 2000 node graphs. In this section we detail a number of simple optimisations that improve running time without changing the core algorithm.

The first optimisation is to reduce the size of the coocurrence vectors. Only including nodes in the vector that actually occur in a coocurrance is a useful optimisation. Both the constraint matrix and the constraint vector depend on the size of the indicator vector for the edges, θ. The size of this vector can be greatly reduced, and hence the size of the constraint matrices as well, by finding the set of edges that could be active,

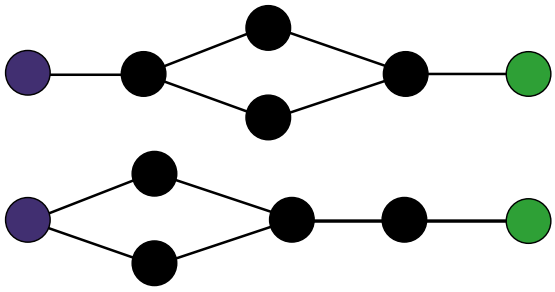


Figure 3.6: An example of two reconstructions that could obey the weak degree constraints and the connectivity constraints for a coocurrence, yet are clearly not paths, and so are not consistent with the coocurrence data

using a naive estimate such as (for coocurrence sets X_i):

$$\bigcup_{i=1..k} \{(u,v)|u \in X_i, v \in X_i\}$$

We can then only include these edges in θ, the rows of the constraint matrix and constraint vector. In practice this greatly reduces the size of θ, usually to the level where the size of the original graph is no longer a factor, just the size of the coocurrences.

3.7 Constraint generation

As previously noted, solving the BIP problem using the full set of connectivity constraints can be computationally prohibitive for larger coocurrences. The practical upper limit for each coocurrence is around 12, any larger and the number of connectivity constraints slows the BIP solver down significantly. For networks where long path lengths occur, another solution is needed. In this section we describe a constraint generation scheme, whereby only the connectivity constraints that are found to be necessary are introduced.

Constraint generation is a high-level methodology for solving integer and linear programming problems with large numbers of constraints. The basic method consists involves repeatedly solving the optimisation problem, starting with only a minimal subset of the full set of constraints. At each step the solution found is checked to see if any of the constraints that were not used are violated. Those that are violated are added to the subset used, and the optimisation is rerun.

In order for constraint generation to be used, it must be possible to determine the violated constraints efficiently. A method for doing so is typically called a strong separation oracle (Grötschel et al. [1988]). There are also practical considerations; if the number of optimisation runs that end up being required is large then nothing is gained by using constraint generation.

It turns out that the BIP formulation described in this chapter is well suited to constraint generation. The initial set of constraints are the degree constraints. Suppose a BIP solver is used with these set of constraints, returning a network G as the solution. For each coocurrence X_i, a depth first search is run on the subgraph of G induced by X_i, starting at the source node. Let C_i be the subset of X_i reached during the depth first search. If $C_i = X_i$ then the coocurrence is connected, and no constraints need to be generated for it. Otherwise, the following constraint is added to the constraint set:

$$\sum_{u \in C_i} \sum_{v \in X_i - C_i} x_{u \to v} \geq 1$$

This is just the connectivity constraint for partition $(C_i, X_i - C_i)$. Once such constraints have been formed for all coocurrences, the optimisation is rerun as described previously.

The number of runs of constraint generation required depends heavily on the path length distribution. In section 4.6 an empirical analysis of connectivity constraint violations is detailed. For efficient implementation of this method it helps to use a BIP solver that can be restarted at the previous feasible solution before the introduction of the extra constraints. However, we found that this was not necessary for our tests, as performance was fast even when starting each optimisation anew.

Chapter 4

Synthetic Test Results

4.1 Test procedure

A number of synthetic tests were performed on a variety of random graph models. Firstly, disjoint sets of source and destination nodes were chosen for each run. The number of sources was fixed at 5 and the number of destinations was varied. Coocurrences were generated between each source and each destination, so that a run with 5 sources and 20 destinations had a total of 100 coocurrence samples. By varying the number of destinations, the number of coocurrence samples is varied, and indirectly the complexity of the reconstruction problem. Each coocurrence was generated using shortest path deterministic routing.

The reasoning behind using source and destination sets of the above form is two-fold. Firstly, it mirrors typical sampling in real world networks, when samples are created by active probing of the network. Secondly it ensures that there is significant overlap between the coocurrence samples.

A new random graph was generated for each run, where a run involved generating a set of coocurrences, then reconstructing a graph from the coocurrence using each of the methods. In this way each method was run against the sample problem. When the random graph generated was not connected, the graph was scraped and another graph was generated.

We measured the performance of each method by comparing the reconstructed graph against the clairvoyant reconstruction. So the reported error does not include missing edges due to that edge never being part of a coocurrence. The error metric used is the *symmetric error*, which is calculated between two sets of edges A and B as the number of edges in A but not in B, plus the number of edges in B but not A (i.e $|A - B| + |B - A|$). The symmetric error is a coarse measure for this problem, as it tends to give a larger value than would be expected. For example, if a reconstruction is correct except that a single coocurrence has a pair of nodes whose order is swapped from the correct order, then this would give a symmetric error of 4 (assuming that the coocurrence doesn't overlap with others).

Tests were performed on undirected graphs, and similarly the symmetric error re-

ported is for the undirected edge set. The coocurrences were preprocessed using the node merging techniques described in Section 2.3.1. This accounts for why our reported symmetric error is lower for all methods than for similar graphs in Rabbat et al. [2006].

For the FM method, a sparsest of 10 approach was used. The algorithm was run 10 times, with the sparsest reconstruction used. The FM method is not an inherently random algorithm, however it must somehow determine what order to place nodes in cases where the nodes have equal weight within a certain path. Our approach was to choose randomly, and this is why the algorithm is non-deterministic. The use of the sparsest reconstruction is motivated by Theorem 1, page 21.

For the ML method, we also used a sparsest of 10 approach. Rabbat et al. [2006] suggests that the reconstruction with the maximum marginal likelihood be used, but in practice we found that there would often be several reconstructions with equal maximum, and that choosing the sparsest was a good way to distinguish them. For these tests we had a known source and destination for each coocurrence, and so the method is simplified from that described in Rabbat et al. [2006]. Namely we did not track the starting probabilities π for each node. We also used the full E step rather than one of the approximations described, so as to get the best results possible.

It was found that the initial values for the transition matrix A had a great effect on the quality of the maximum returned. The EM method used to solve the ML solution to the coocurrence problem only returns a local maximum, and the quality of this result depends heavily in the starting point in the search space. An *i.i.d.* valued random matrix was tried as the initial value of A but performance was poor. The runs presented here were done using an initial matrix A containing values 0.4, with each value perturbed by ± 0.2, and normalised so that each row summed to 1.

4.2 Discussion of results

- In the case of Erdos-Renyi and Watts-Strogatz random graphs, the BIP method out performs existing algorithms, see Figures 4.1 & 4.2. The difference in performance between the BIP method and the weak BIP method is negligible, so not much is lost using the weakened set of constraints. Likewise the improvement over the ML method is small but consistent. In any case, the error rate is very small compared to the number of edges in the reconstruction, which is between 50 and 80 depending on the instance.

4.3 Implementation & running time

We implemented all three algorithms in matlab, without using any native code optimisations. The FM & ML methods did not require any outside libraries. The BIP method requires an integer programming solver. Initially we used matlab's `bintprog`

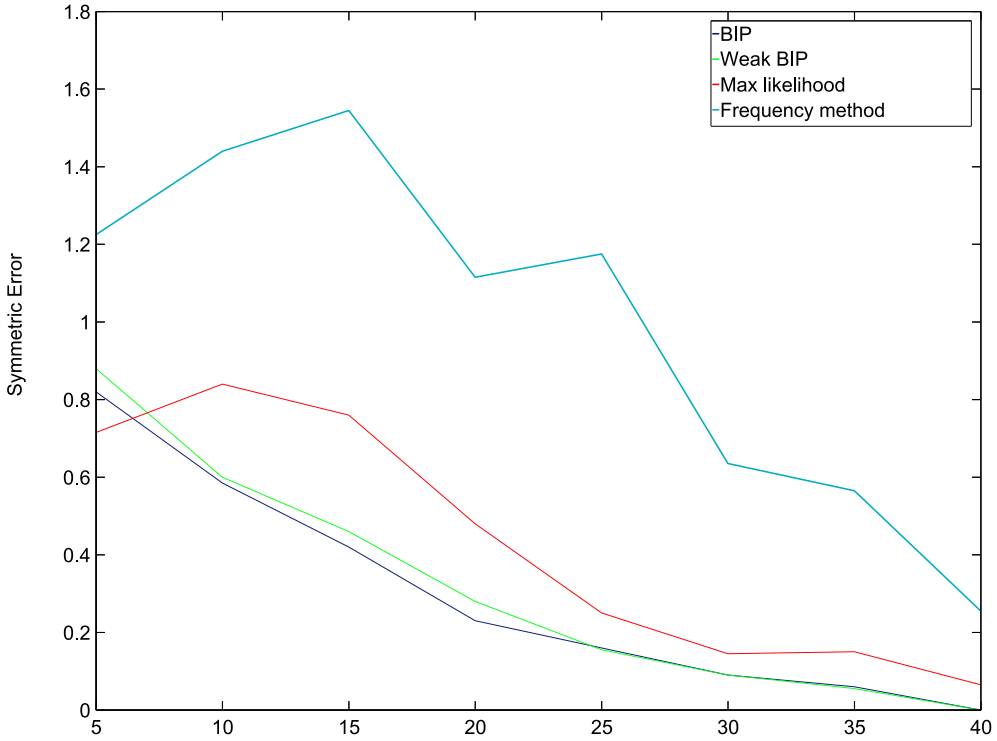


Figure 4.1: Performance of each method on Erdos-Renyi random graphs

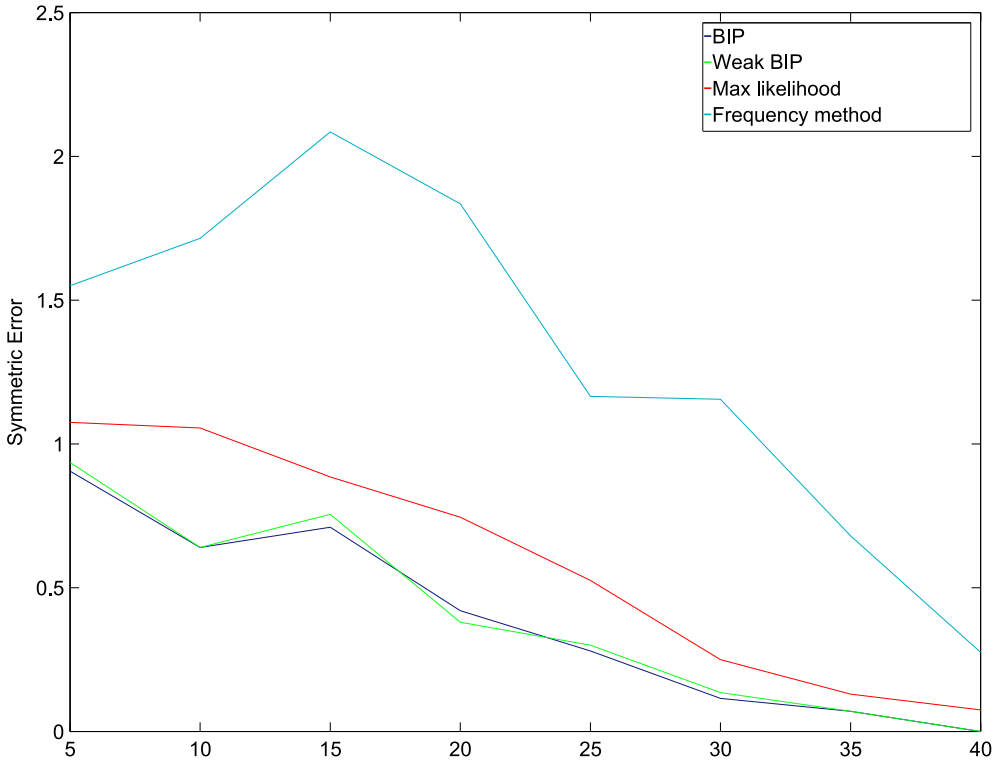


Figure 4.2: Performance of each method on Watts-Strogatz random graphs

Table 4.1: Scaling of running time in seconds as graph size increases, when number of coocurrences is fixed

Method/graph size	50	100	200	500	1000	2000
FM method	0.078	0.092	0.100	0.139	0.237	0.618
Naive BIP impl.	0.160	0.330	0.710	2.68	7.741	25.544
Efficient BIP impl.	0.109	0.166	0.207	0.279	0.299	0.330

from the *Optimisation Toolbox*. This was fast, although unfortunately it could not handle many of the constraint matrices we used, failing with errors relating to the constraints not being feasible (Even though they were). The `lp_solve` (Berkelaar et al. [2010]) library had none of these problems, and performed equally fast. All the results presented here were computing using it. We don't present running time figures for the EM method here, as our implementation was unoptimised, and so would not be indicative of the performance of a good implementation.

To get an idea of the running time of our algorithm compared to the FM method, we conducted a series of tests. We ran each method on the same randomly generated coocurrences & graphs, as in the previous section. 50 runs were completed, for a variety of graph sizes. The results are shown in Table 4.1. For the BIP method we used both degree & connectivity constraints, and used a sparse version of the algorithm. Unidentifiable node set merging was used for both methods. The figure shows results for both an unoptimised and optimised version of the BIP method. This shows the the necessity of an efficient implementation, when testing against large graphs. The optimisations are detailed in Section 3.6.

Our test machine had an Intel i5 750 processor running at 2.8 *Ghz*, and 4 *GB* of memory. The density of the graphs was increased as the graph size was increased, to ensure that no more than 4 graphs needed to be generated each time, to ensure at least one was connected.

The running time of the FM method appears to scale linearly with the size of the graph. The $O(N_i \log N_i)$ theoretical term for each path due to the sorting seems to be negligible compared to other aspects of the implementation. The BIP method's running time is not clear theoretically, as it relies on the Simplex algorithm, whose running time is polynomial for most inputs, but can be exponential in pathological cases. The unoptimised methods running time seems to scale by 2-4x when the number of nodes is doubled, suggesting a low degree polynomial running time, probably due to overheads in the sparse matrix computations. The optimised version scales roughly linearly, like the FM method.

Note that for the implementation of the BIP method, we used sparse vectors representing the full set of edges for the graphs of each size. In theory the overhead from the larger sparse matrices should be small, but in practice for the BIP engine we used there was significant overhead. This is part of the reason why the optimised version is quicker.

The test above gives an idea of the effect of the graph size on reconstruction performance, but it only measures the effect of path length on reconstruction performance indirectly. The average path length only scaled slowly between the small and large graphs in that test. As the maximum and the average path length are the main contributing factor to the running time, additional tests were run to isolate those factors.

A series of random (euclidean) geometric graphs were used for the next test (see Section 5.7). Geometric graphs have high average path length, and they allow us to control this path length. By generating the graph nodes within a rectangular region with the shorter side a fixed length, we can keep the density close to constant, while adjusting the average and maximum path lengths by changing the longer side's length. We performed the test with and without the connectivity constraints. The use of the connectivity constraints is discussed in Section 4.6. The number of nodes used was varied proportionally with the side length, so that the expected number of nodes per unit area remained the same for all tests.

Table 4.2 shows the results. We firstly note that the average path length increases by about 1 for every unit added to the region length. Doubling the region area does not double the average path length. This is largely due to the merging of unidentifiable node sets. The distribution and maximum path length is also interesting, see Figure 4.3. It varies between 7 for the square region, to 20 for the 4 by 1 region. The distribution changes from near symmetrical for the square region to a fat tailed distribution for the rectangular regions.

The average number of edges in the graph does vary linearly, as would be expected due to the constant spatial density. In each case the proportion of nodes witnessed by the coocurrences is similar although the proportion of nodes that the coocurrences pass through is not. This suggests the formation of *hubs* (See Section 5.5) is occurring, so that many of the nodes only conduct coocurrence signals emanating from themselves.

The running time of the two methods scales similarly to the average path length. The FM method would be expected to scale near linearly with the average path length (although it sorts the paths, which is $O(n \log n)$, the log factor is negligible when sorting such small lists), and this does indeed occur. The running time of the BIP method is 46% slower for the square region graphs, and the margin increases to 216% for the longest region. The BIP method running time is not linear in the path length, although it scales in such a way that even long path lengths are practical.

4.4 Sampling errors

In many real world cases there is a small chance of errors when measuring coocurrences. These might occur in the form of missing or extraneous nodes in a fraction of the coocurrence samples. To assess the effect this has on reconstruction performance, we produced coocurrences samples in the same way as for the previous section, but

Table 4.2: Running time and associated coocurrence statistics, for varying geometric graph regions, with a shorter side of unit length

Region long size length	1	2	3	4
Nodes in graph	50	100	150	200
Avg. path length	3.59	4.56	5.78	6.72
Avg. Edges in graph	263.9	569.1	882.5	1193
Avg. Edges witnessed	161.8	334.4	555.0	717.1
Avg. Nodes witnessed	33.9	53.5	69.1	77.7
FM method time	0.065	0.102	0.128	0.160
LP method time	0.095	0.168	0.248	0.346
BIP method time	0.100	0.187	0.264	0.348
Percentage consistent	99%	95%	90%	68%
Constraints generated	1	1	1.35	1.68

we perturbed each coocurrence with probability p. Each perturbation was chosen with equal probability between adding another node to the coocurrence, or removing an existing one. Figure 4.4 shows the effect for various p.

The first thing to note is that the symmetric error of the reconstruction appears to scale linearly with the probability of perturbation. This is reasonable; if we perturb twice as many coocurrences we would expect twice as many reconstruction errors. The figure shows the results for the FM method in green, the weak BIP method in blue and the ML method in red. The standard BIP method can not be used, as the perturbations introduce violations of the shortcut free property.

We see that both the ML and weak BIP methods perform consistently better than the FM method, under any reasonable amount of perturbation. Interestingly, the weak BIP method maintains its advantage over the ML method only for small amounts of perturbation. After 2.5% the ML starts to perform better, and by 9% perturbations the ML method is performing noticeably better, whereas the weak BIP method is only matching the FM methods performance. So for networks where sampling involves a chance of incorrect samples, the EM method is the best choice. Only if the chance of sampling errors is minuscule is the weak BIP method better. It should be noted that for significant amounts of sampling errors ($>$ 10% samples corrupted) that the relative error becomes similar for each of the three methods. In those cases it might be better to use the FM method due to its simplicity and fast running time.

4.5 Random routing

While the unit-length edge shortest path routing scheme used in the comparisons above is often a good approximation to real work routing, sometimes radically different routing is evidently occuring. In this section we discuss two alternate rounding schemes, and compare each reconstruction method under those schemes.

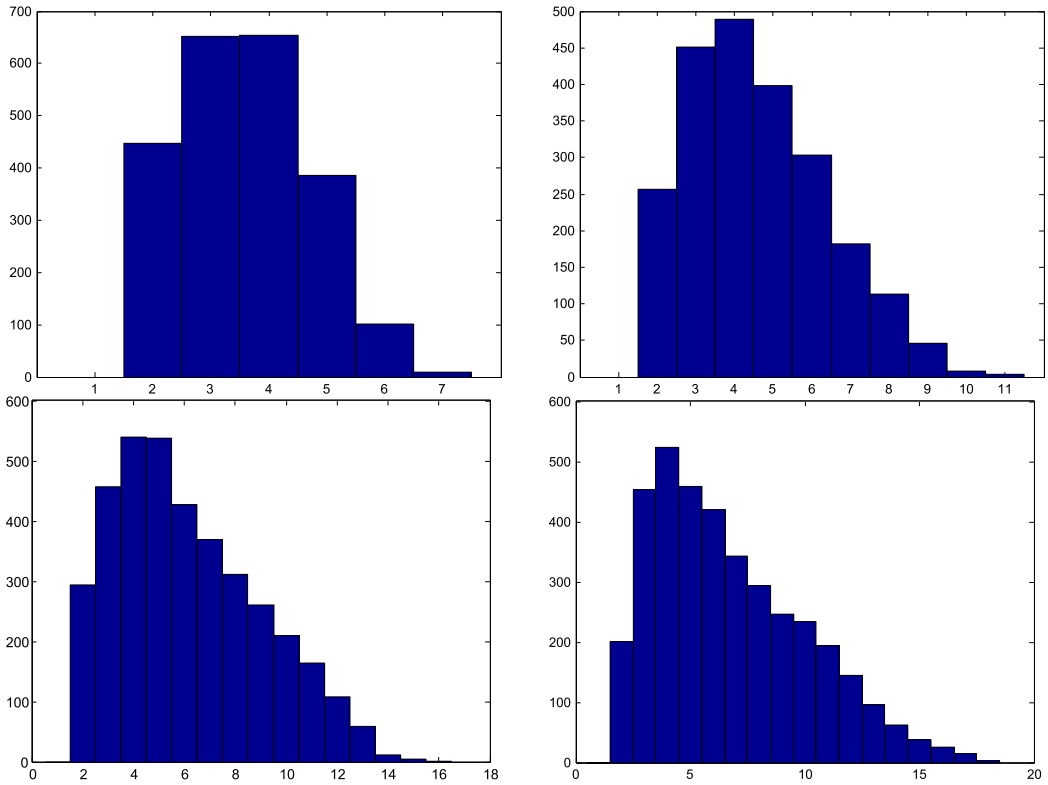


Figure 4.3: Path length distributions for 1 by 1,2,3,4 regions

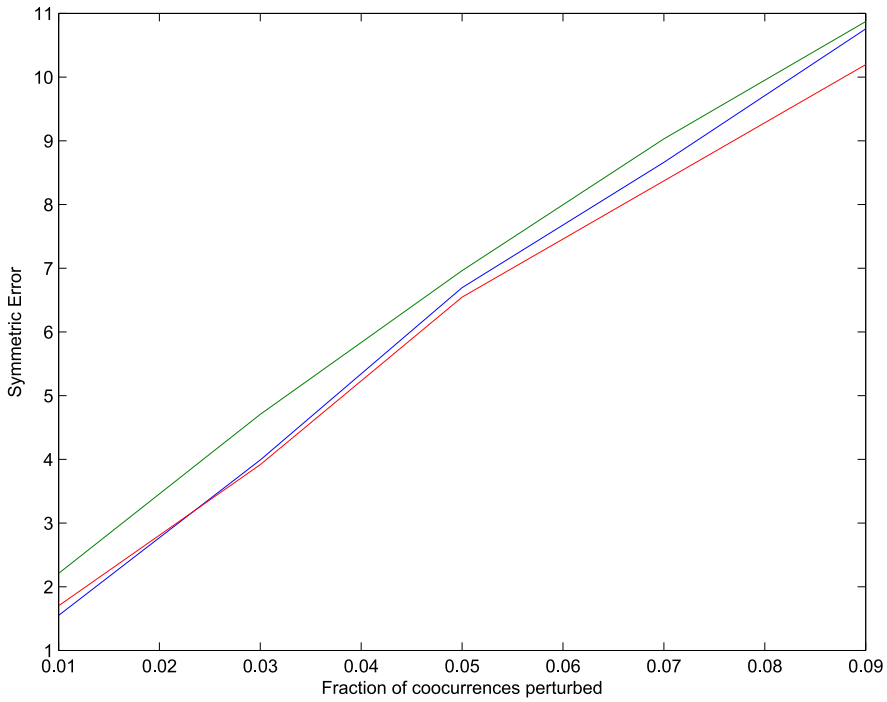


Figure 4.4: Symmetric error for randomly perturbed coocurrences. FM method is shown in green, the weak BIP method in blue and the ML method in red.

The simplest extension of unit-length edge routing is random length edge routing. A shortest path scheme on such a weighted graph has a number of properties that make it an interesting test. Random edge weights do not obey a triangle inequality, so they have less structure than the unit-edge case, and in particular the set of coocurrences generated is not guaranteed to be shortcut free. The number of nodes in each path will be on average higher, making reconstruction more difficult as well. Due to the obvious similarities to the unit-length case, the performance is not expected to be radically different. This is perhaps the best test of each method performance on networks without the shortcut free property, which still have some sort of regular routing scheme.

The other case we consider is random walk routing. This consists of paths generated by starting at a randomly chosen node, and 'walking' by a Markov process, to form a path. More precisely, signals are routed from one node to the next using a purely local probability distribution over that node's neighbours. Unlike the random weight case, this routing is not deterministic. Deterministic routing was a crucial assumption made in proving the effectiveness of the BIP method, so determining if it is similarly crucial in practice is important. For our tests, we sampled the path lengths from a binomial distribution (n=8, p=0.25), and then performed a random walk of that length, starting at a new, randomly chosen source each time. When a walk could not be completed without repeating nodes, it was ended early. The probability distribution used for each node was the uniform distribution over its neighbours.

Figure 4.5 shows the average symmetric error under the different reconstruction methods for random weight routed coocurrences. Note firstly that the FM method does much worse than the other methods, compared to when shortest path routing is used. Also note that the ML method performs consistently better than the BIP method, by a factor of as much as half the error. Interestingly, the error is not effected by changing the number of coocurrences; it stays largely the same when the number of destinations are varied, under all but the FM reconstruction method.

Figure 4.6 shows the error for random walk coocurrences under the same graph model. There is a larger separation between the different reconstruction methods than in the other cases that have been considered. The LP method performs best here, by 20-50% margin across the range. Unlike the random weight case, the error rate does go down gradually as the number of coocurrences is increased, except when FM reconstruction is used. The FM method shows the opposite effect; the error rate becomes higher as the number of coocurrences is increased. Also note that the error rate is higher across the board than for random weight and shortest path routing. It seems that random walk coocurrences give less information that aids reconstruction. The error rate is still much better than chance, clearly the consistent routing probabilities are helping the reconstruction.

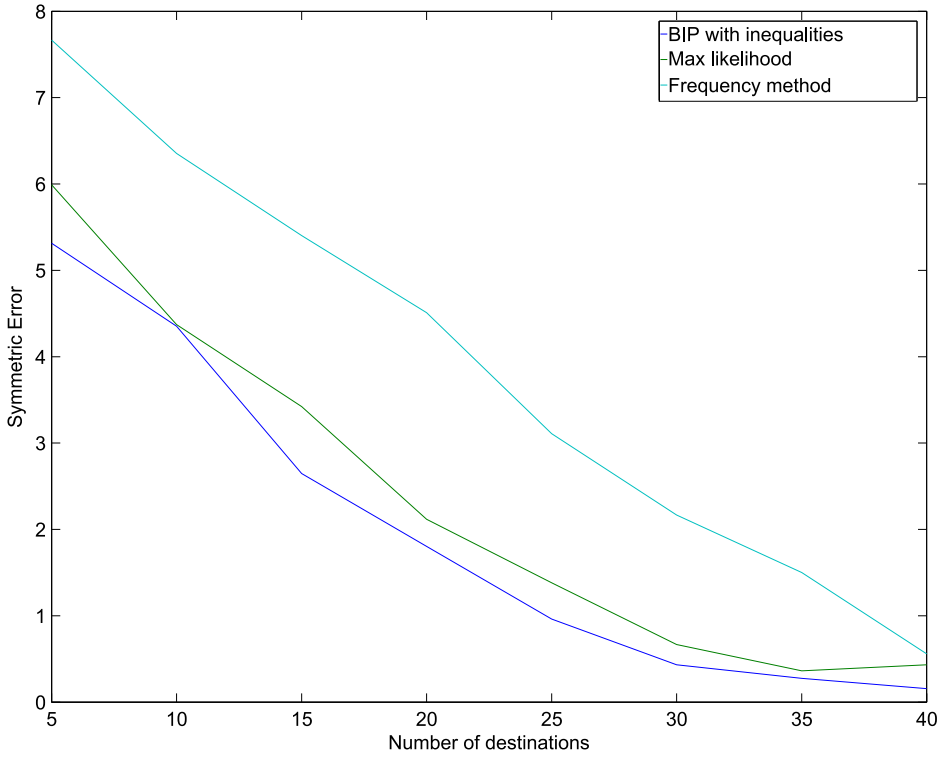


Figure 4.5: Symmetric error for random weight coocurrences.

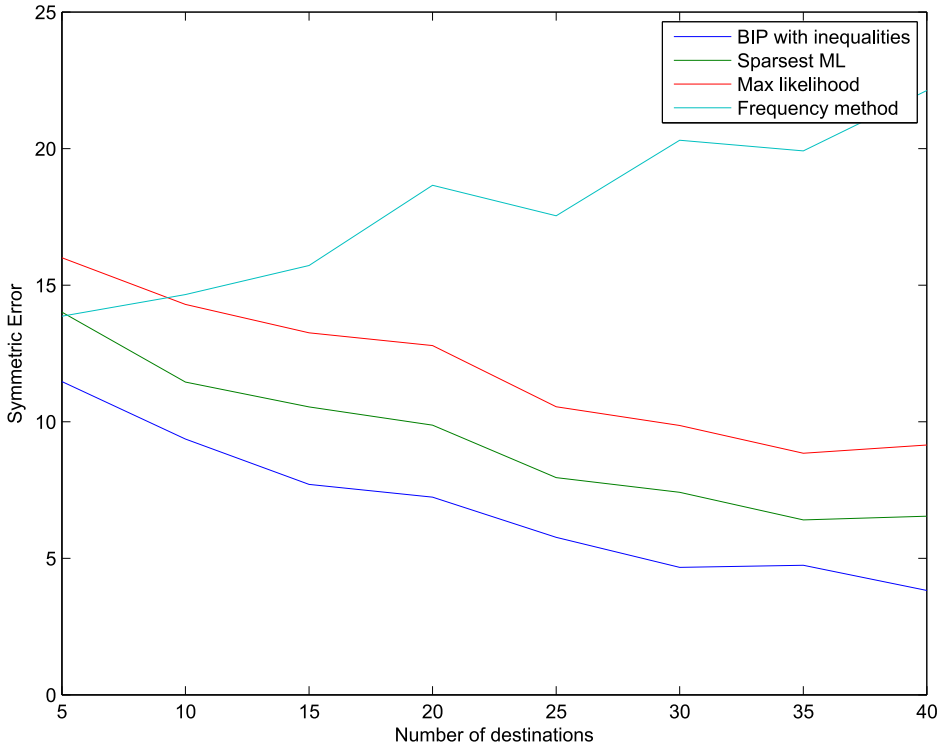


Figure 4.6: Symmetric error for random walk coocurrences.

4.6 Connectivity constraint violations & non-integer solutions of the BIP method

In Section 3.2 two types of linear constraints were described that together enforce that the solution recovered using the BIP method is consistent with the coocurrance data. While the number of degree constraints required is linear in the lengths of the paths, the number connectivity constraints required scales exponentially. It was discovered during development of the BIP method that the connectivity constraints were rarely necessary. In the majority of cases the reconstruction found using just the degree constraints was consistant. In order to investigate this further we performed a number of tests. For the 50 node Erdos-Renyi graphs it was found that about 1 in 10,000 reconstructions were not consistent. This percentage of runs could not be pinned down exactly due to large number of runs that would be required.

To better quantify the probability of this occurring, tests were performed on the same random geometric graphs as used in Section 4.3. It was expected that the much longer path lengths would result in a higher probability of the reconstruction being inconsistent. Paths that obey the degree constraint but not the connectedness constraint must contain at least 5 nodes, and only minimally overlap other sampled paths (see Figure 3.3). The results of this test are shown in Table 4.2. For square regions, 99% of the runs resulted in consistent reconstructions. This drops down to only 68% for the four by one region. In problem instances with such long path lengths, a constraint generation method needs to be used, to gradually introduce the violated connectivity constraints. It should be noted that these geometric graphs are not what is encountered in most applications. In Chapter 5 we discuss more typical random graph structures, and a variety of real world networks. On those real world networks the connectivity constraints were not necessary for consistent reconstruction during our tests.

It was also noted in Section 3.2 that when using a BIP solver that uses LP relaxation internally, normally the initial LP relaxation was integer. So a branch and bound method almost always solved the problem without any branching. To quantify this, a number of tests were performed. The main test was an empirical comparison of running time between a pure LP solve and a BIP solve, using the geometric graphs as described in Section 4.3. Table 4.2 gives the running times. Note that the running time difference between the two methods is 5-10%, and generally negligible. This shows that very few branches were required, as the BIP algorithm used solved a LP instance at each node of the search tree. Note that the BIP & LP solver were both part of the lp_solve software, so no performance difference was introduced by using more efficient software. For more reasonable classes of graphs, such as the Erdos-Renyi model, non-integer solutions did not occur in practice on 50 node networks, not even when a batch of 10,000 runs was performed.

Table 4.2 also shows how effective a constraint generation scheme can be. The *constraints generated* row shows how many constraint generation cycles were need for those instances for which using only the degree constraints resulted in an inconsistent

reconstruction. This value appears to grow slowly, so that least for these geometric random graphs, the constraint generation method allows for solving much larger instances than is possible using all of the connectivity constraints.

Chapter 5

Realistic Network Models and Real World Networks

In previous chapters we explored methods for coocurrence reconstruction without regard to the structure of the underlying network. We showed that under certain assumptions about the routing of signals through the network, it was possible to do better than with existing methods. The empirical results for the synthetic networks tested against in Chapter 4 showed good performance when coocurrences were generated on simplistic random graphs. In this chapter we give an overview of existing results about the structure of real world networks, including a number of random graph models thought to approximate certain classes of random networks. We also give empirical results for the error in concurrence reconstruction on these network models, as well as for a selection of real world networks.

Random graph models are particularly interesting, as they can display the same properties as real world graphs, while allowing us to generate large numbers of sample graphs to test the performance of the concurrence algorithms on. For experiments on real world graphs we are limited to a few different examples, and to typically very large graphs, which makes large numbers of random tests computationally prohibitive.

Real world networks can be measured by two generally orthogonal properties, the average path length (Section 5.1) and the clustering coefficient (Section 5.2). We discuss these properties and show what effect they have on reconstruction performance.

5.1 The small world phenomenon

Almost everybody is familiar with the small world phenomenon in some way. It can be though of as the observation that people of diverse backgrounds, locales and ethnicities are often linked through a small chain of friendships to one another. This seems to be a general principle of networks between people, not just social networks. The idea gained traction with the general public as a game, where people would attempt to link a randomly chosen actor to another actor (usually Kevin Bacon) through

a series of co-starings in movies. In both the game and in social networks it usually takes less than 6 edges, and often only 2 or 3. Many other networks also display this property, and indeed most random graph models do also. Any graph exhibiting the small world property is called a *small world graph*.

The most common measure of the degree to which a graph is small world is the average path length. This is measured as the average length of a path generated by choosing two nodes randomly, then finding the shortest path between them. Graphs for which the average path length is of the order of $\ln n$ are considered small world. The terminology *ultrasmall* has been used to describe networks for which the average path length scales as $\ln \ln n$, or is constant. Other measures include the *diameter* of the graph, which is the maximum length out of all the shortest paths between pairs of nodes, and the 90th percentile path length, which is the 90th percentile of the distribution of shortest path lengths between randomly chosen pairs of nodes.

The small world property is not universal, as networks relying on physical or temporal proximity do not usually obey it, especially when large spans of time or distance are used. One example would be social networks involving people not currently alive, as it would obviously require a large number of steps to link some one from ancient Egypt to a modern individual. Similarly the structure of a wireless mesh network would also require a large number of steps to link nodes at opposite ends of the network.

The number of possible reconstructions for a set of coocurrences depends directly on the length of each coocurrence. When signals are routed using shortest-path routing, this becomes a direct dependence on the average path length. However, this doesn't necessarily tell us about the degree of overlap between the coocurrences, which is what determines the number of 'good' solutions.

5.2 Clusters & the clustering coefficient

Another property typically found in real world networks is clustering. This is the tendency for nodes to form groups in which the number of edges in the group are much higher than the average for the rest of the graph. This is another phenomenon most easily recognised in social networks, where the groups correspond to cliques (in the social sense). As opposed to the global nature of the small world property, this can be described as a local property. There are a number of related measures of clustering, depending on whether one wants to measure the clustering at a particular node, or aggregate for the whole graph.

The global clustering coefficient measures the average clustering over triples of nodes. More formally, we consider the ratio of closed triplets (3-cliques) to all triplets. A triplet is defined as three nodes directly connected in the sense that there is either 2 or 3 edges between them. A closed triplet is simply the 3 edge case.

The local clustering coefficient measures clustering at a particular node. This can be

used in aggregate as well, by considering the average local clustering coefficient. The local clustering coefficient of a node v is defined as the ratio of the number of edges in the subgraph formed by the nodes adjacent to v to the number of edges in the complete graph with the same number of nodes. This essentially is a measure of how close v and its neighbourhood is to being a clique in the graph sense.

5.3 The Erdős-Rényi model

The Erdős-Rényi model is the first random graph model to have been studied in detail (Bollobas [1985]). Named after Paul Erdős and Alfréd Rényi, it has a number of interesting properties, despite its simple construction. An Erdős-Rényi graph $G(n, p)$ is constructed as a graph of n nodes, where starting with an edgeless graph an edge is added between each pair of nodes with probability p. The number of edges are obviously binomially distributed $B(n, p)$, as each edge is independent. The other properties are more interesting. The average path length between two randomly chosen nodes is proportional to $\ln n$, so Erdős-Rényi graphs exhibit the small world property. However, unlike most real world networks, the clustering coefficient is typically small. This model may be regarded as a unrealistic, as very few real world networks have edges attached independently in this way. However, it forms a good baseline for comparison, as given the lack of structure we might expect this to be a worse case for our set of algorithms.

We generated 100 graphs, using 50 nodes and $p = \frac{4}{n} = 0.08$. This gives a high probability of the graph being connected, see Bollobas [1985] for results about the sharp threshold for connectivity for Erdős-Rényi graphs. This probability was chosen so that the expected number of edges is roughly 100. Runs were done using 75 coocurrences. This corresponds the the worst case performance, found when varying the number of coocurrences. The average number of edges witnessed by the coocurrences was 60, and ranged from 50 to 72.

The number of indistinguishable nodes sets was very small. On average only 0.6 merges were performed per run. The path lengths were not greatly modified by this merging; the graph on the right side of Figure 5.2 is almost identical to the one on the left. The merging still reduces the number of solutions a great deal, essentially eliminating the outliers (see Figure 5.1).

The distribution of the symmetric error in terms of the number of destinations shows an interesting 'hump' around 12 destinations, where the error is significantly higher than for fewer destinations. It is clear from Figure 5.3 that as the number of destinations increases to a significant portion of the total graph, the error rate drops towards zero. This is due to the overlap between coocurrences increasing to the point where reconstruction is easy. At this point the coverage of the graph's edges by the coocurrences is at nearly 90%, so the reconstruction even gives an large proportion of the underlying graph. The cause of the hump is not clear. It may be because the chance of a long path increases with the number of destinations (as long paths are outliers),

and this effect is counter-acting the effect of having more coocurrences (which as mentioned makes the reconstruction easier due to overlap).

5.4 The Watts-Strogatz random graph model

The Watts-Strogatz model (Watts and Strogatz [1998]) is perhaps the canonical example of a small-world network. Unlike the ER model discussed previously, it exhibits both short average path length and high clustering coefficient. Watts & Strogatz's key idea was to consider the two canonical examples for the two properties, and to find a way of interpolating between them such that the resulting model had properties of both. The two models they used were the ER model, and a regular ring lattice.

As discussed previously, the ER model has a short average path length that scales logarithmically with the size of the graph. A regular ring lattice is a not a random graph model, but rather a simple graph construction. It is a ring of nodes, each joined to k of their neighbours. Such a structure has a clustering coefficient of $\frac{3}{4}$ when k=4, which is very high for a graph with a small number of edges. See Figure 5.4 for a diagram of the $k = 4$ case.

A regular ring lattice has a very high average path length, which scales linearly with the number of nodes in the network. Watts & Strogatz found that by randomly rewiring a small number of edges, the average path length was greatly reduced, without significantly effecting the clustering coefficient. The reasoning is intuitive; moving a few edges in the graph only effects the clustering coefficient slightly, in an effectively linear way, whereas the effect on average path lengths is highly non linear. Just one rewired edge can reduce the path length between previously disparate parts of the graph.

They used an interpolation parameter β, which is just the probability of each edge being rewired. $\beta = 0$ corresponds to the regular ring lattice, and $\beta = 1$ to ER random graph. Edges are rewired in the following way. The set of nodes are looped over, and for each node v each edge e incident to v is chosen with probability β. If an edge is chosen, then it remains incident to v at one end, but the other end is attached to another node in the graph, chosen uniformly from the set of nodes that are not already adjacent to v. This excludes loops ($v \to v$) and duplicate edges.

A Watts-Strogatz random graph still resembles a ring lattice when a small β parameter is used, yet its properties resemble real world graphs well. Few real world graphs resemble ring lattices, so this is a good example for how the two properties discussed so far are not sufficient to completely characterise real world networks. It still provides a good test to gauge the effectiveness of coocurrence reconstruction on graphs with these properties.

The regular ring lattice of 50 nodes, where each node is joined to its 4 nearest neighbours has 100 edges, and $\beta = 0.5$ is known to maintain the structure of both the ER

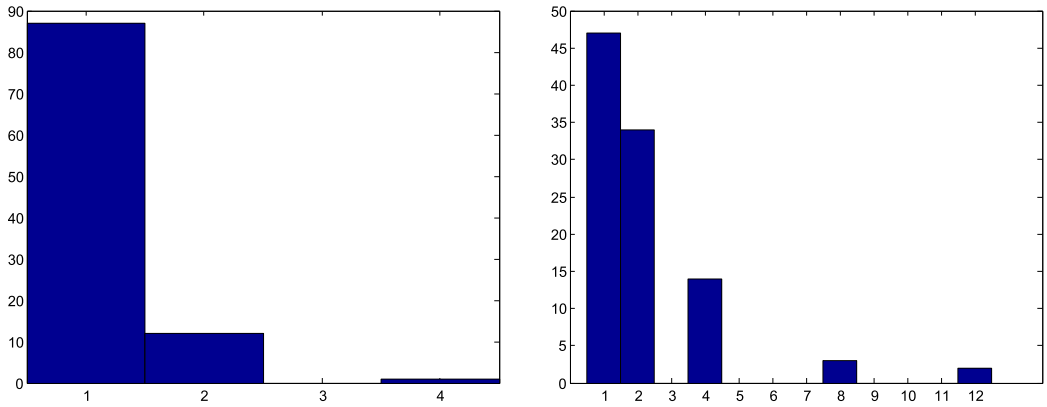


Figure 5.1: Number of solutions for each run for Erdős-Rényi random graphs. Runs without merging of indistinguishable node sets are on the right.

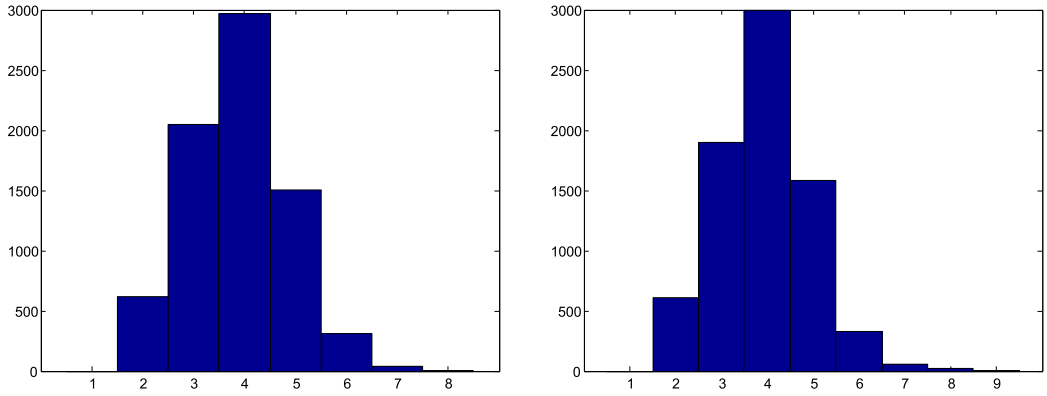


Figure 5.2: Path length for coocurrences generated on Erdős-Rényi random graphs

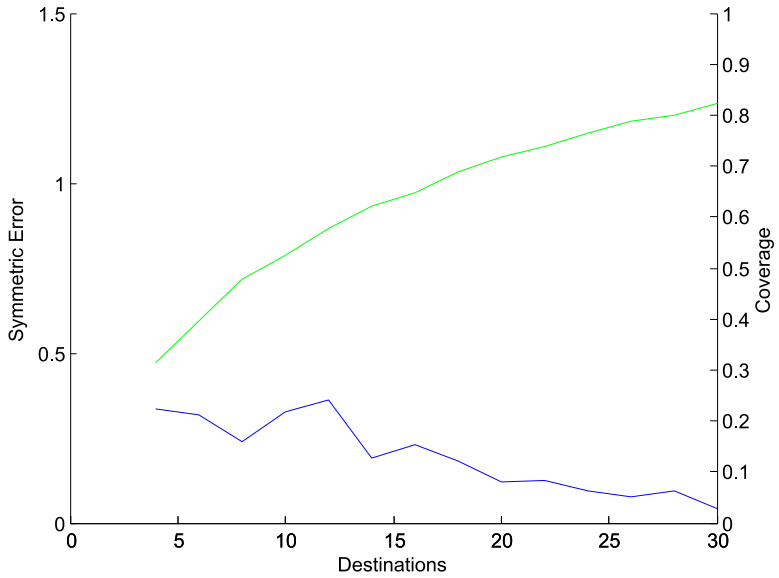


Figure 5.3: Coverage & error rate as number of coocurrence samples is increased for Erdős-Rényi random graphs

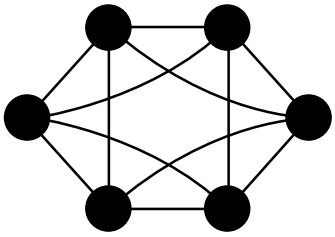


Figure 5.4: A regular ring lattice with 6 nodes, and $k = 4$

and ring lattice. Using such a model, we ran the same tests as for the ER model, so that the results could be directly compared.

The first thing to note is that the average path length is similar to that of the ER model, as expected. For both the path length distribution has a mode of 4. See Figure 5.6. The WS model seems to have more than twice the number of runs where there are more than 1 solution (Figure 5.5). This suggests the high clustering has an adverse effect on the number of solutions. There is a direct link between the number of solutions and the error rate for reconstructions, so it is not surprising that the error rate in Figure 5.7 is higher.

When merging of indistinguishable node sets is turned off, the effect is similar towards the number of solutions as for the ER model, just with a corresponding higher number of solutions as in the case with merging. Interestingly, more runs have 2 sparsest solutions than a single solution, which is a quite different result from the ER case.

The change in coverage is similar to the ER case, just with about 5% less coverage for any given number of samples, see Figure 5.7. This could be explained by the slightly lower average path length.

5.5 Scale free networks & preferential attachment

There is another prominent property of real world networks, the tendency of *hubs* to form. A hub is just a node with a much higher degree than the average. Real world networks often have hubs. Examples includes the backbone routers of the internet, and Celebrities in social networks. Indeed empirically the degree distribution of nodes in real world networks often follows a power law, a fact originally noted in citation networks (de Solla Price [1965]), but later found to be pervasive. Power law distributions tend to produce outliers, due to the so called *fat tail* property, where outliers are statistically much more likely than for other distributions such as the binomial distribution. These outliers are exactly the hubs observed in practice.

A power law distribution is any of the form

$$P(x) \propto x^{-\gamma}$$

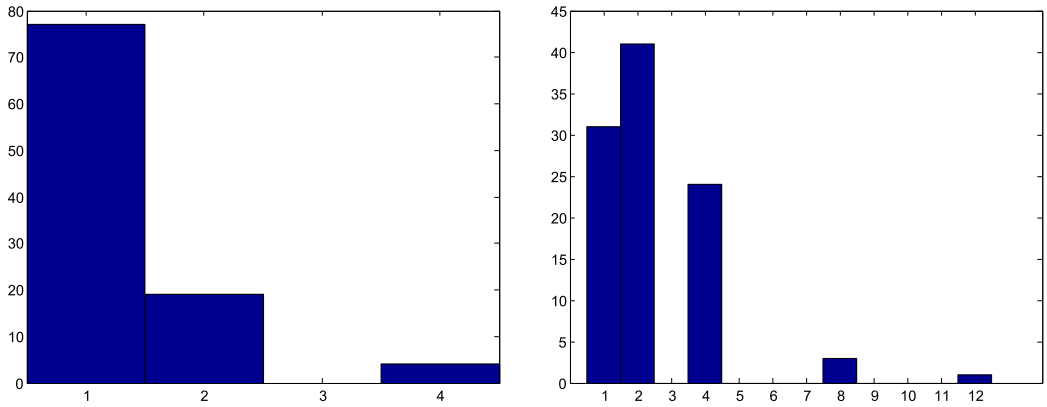


Figure 5.5: Number of solutions for each run for Watts-Strogatz random graphs. Runs without merging of indistinguishable node sets are on the right.

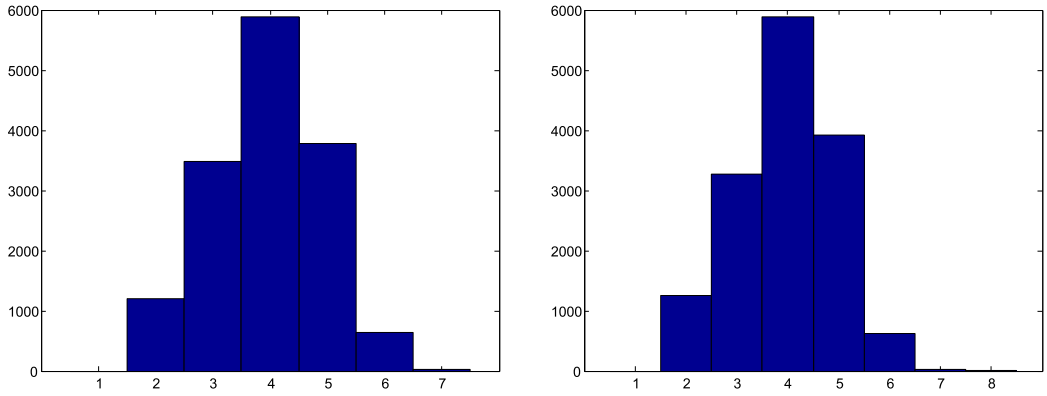


Figure 5.6: Path length for coocurrences generated on Watts-Strogatz random graphs

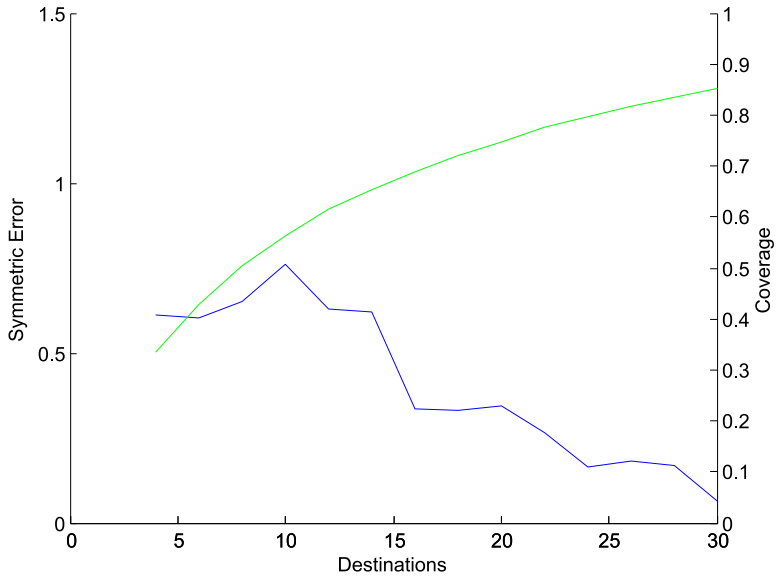


Figure 5.7: Coverage & error rate as number of coocurrence samples is increased for Watts-Strogatz random graphs

Where γ is a constant, usually in the range $[2, 3]$. Networks where the degree distribution follows such a power law are called *scale-free networks*.

As the existence of hubs is explained by a power law degree distribution, much research has focused on finding mechanisms for the formation of networks that result in such degree distributions. For networks formed gradually, perhaps *organically* is a better word, the *preferential attachment* mechanism is the most well known.

Preferential attachment (Barabasi and Albert [1999]) is a mechanism where newly attached nodes form connections to existing nodes, with probability proportional to the current degree of each existing node. Essentially those nodes with a lot of neighbours are more likely to be connected to the new node than those with few neighbours. So the idea is that the probability of a new node u being connected to an existing node v is *very roughly*

$$P(e_{u \to v}) \propto f(d(v))$$

where $d(v)$ is the degree of v and f is some monotonic function, in practice usually linear.

The property of a network being scale free is not totally independent of the two other network properties discussed. Scale free networks don't have to display high amounts of clustering, although they often do. Numerical simulations have shown that the BA model (Section 5.6), a typical scale free network shows 5 times more clustering than the ER model (Barabasi and Albert [2002]). Scale free networks also show a particularly short average path length, even shorter than for ER random graphs (Barabasi and Albert [2002]). A result by Cohen and Havlin (Cohen and Havlin [2003]) showed that for values of γ in the typical range for applications ($2 < \gamma < 3$), the graph diameter grows only at a rate proportional to $\ln \ln n$. Small deviations from a perfect scale free structure can effect this in practice of course.

5.6 The Barabási-Albert random graph model

The Watts-Strogatz model displays some of the properties of real world networks, but it produces graphs that don't intuitively look like real world graphs. The clusters it produces have to much structure, and it doesn't tend to produce hubs. A more realistic model is sometimes necessary, one where the degree distribution follows a power law.

The Barabási-Albert model (Barabasi and Albert [1999]) is one such model. It directly applies the preferential attachment mechanism to generate a graph by adding one node at a time. Formally the following process is used. A 'bootstrap' graph G_0 is required, with a small number of nodes n_0, to start the process of. This graph should have a minimum degree of 2 for each of its nodes. Then new graphs are generated, $G_1 \ldots G_{n-n_0}$. The graph G_i is formed by taking the graph G_{i-1}, and adding a new node u. A number of edges (m) are added between u and nodes $v \in V_{i-1}$, where V_{i-1} is the

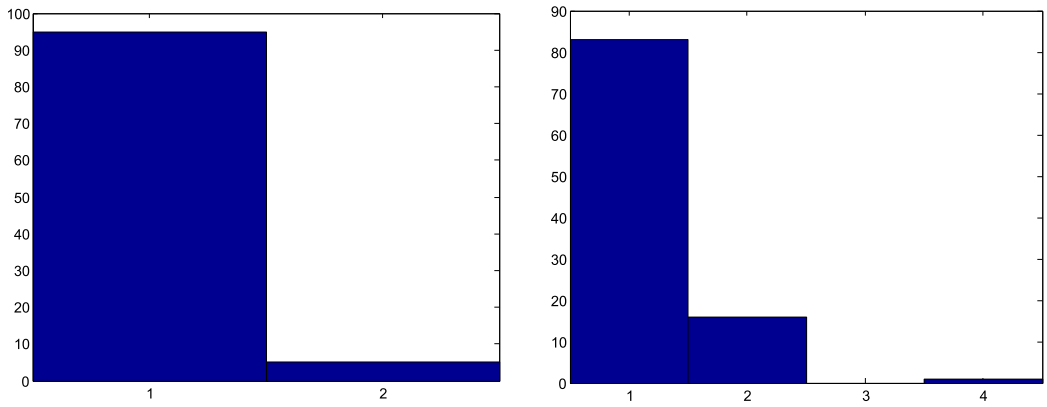


Figure 5.8: Number of solutions for each run for Barabási-Albert random graphs. Runs without merging of indistinguishable node sets are on the right.

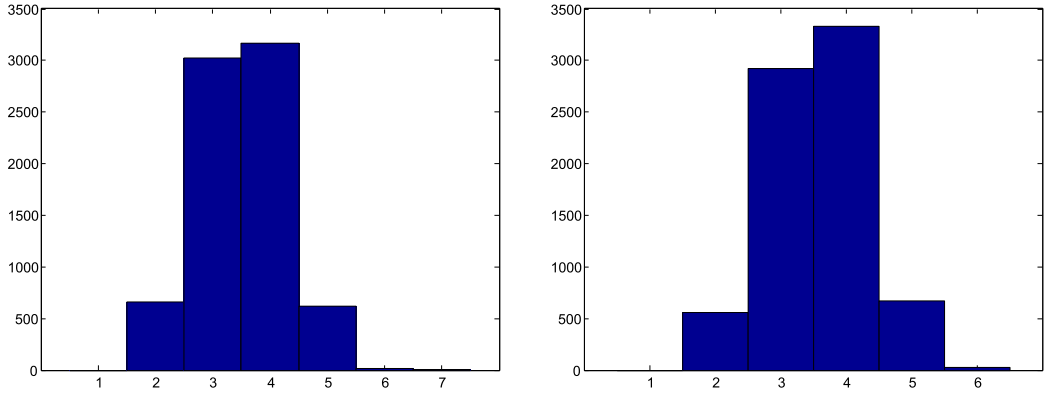


Figure 5.9: Path length for coocurrences generated on Barabási-Albert random graphs

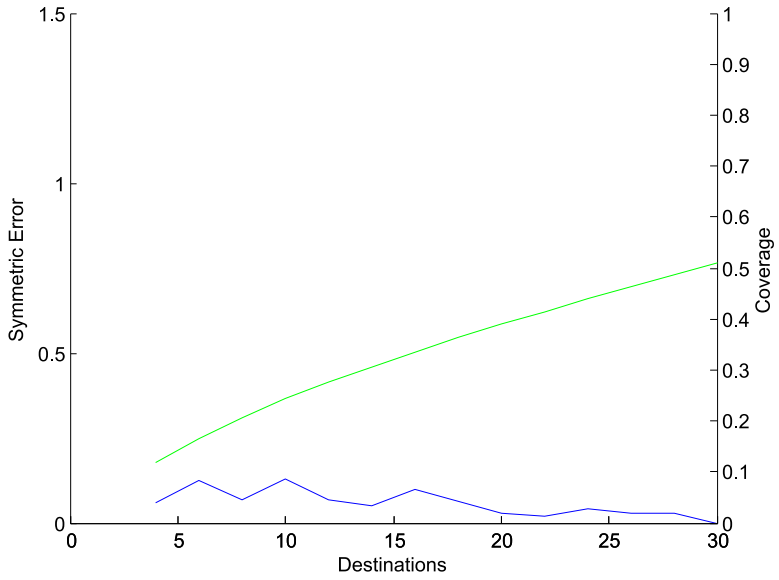


Figure 5.10: Coverage & error rate as number of coocurrence samples is increased for Barabási-Albert random graphs

vertex set of G_{i-1}, by choosing v via sampling from the following distribution over V_{i-1}:

$$P(v) = \frac{d(v)}{\sum_{x \in V_{i-1}} d(x)}$$

The resulting graph $BA(n, m)$ has n nodes, and obviously has $d(v) > m, \forall m$. The bootstrap set of nodes has little effect on the structure for large n, so we don't consider it as a tuneable parameter. It turns out (Barabasi and Albert [2002]) that this model results in a degree distribution that follows a power law, with exponent $\gamma = 3$, independent of the tuneable parameters.

The BA model is another model with short average path length (Barabasi and Albert [2002]), roughly proportional to $\log n$. Its clustering behaviour is more difficult to analyse, the empirical results show that it varies with the size of the graph, following a power law (Barabasi and Albert [2002]). This doesn't fit the definition of a small-world network for all network sizes; for some small n or large m most generated $BA(n, m)$ graphs would be considered small world.

As with the previous two models, we ran tests using randomly generated BA graphs with 50 nodes. Unlike those models, the m parameter in $BA(n, m)$ gives us exact control of the *number* of edges, rather than over the *expected* number of edges. We choose $m = 2$, so that the generated graphs had 100 edges.

Interestingly, the symmetric error for the BA graph reconstruction is much lower than for the other random graph models considered. The error rate stays below 0.1 on average when more than 5 destinations are used, only performing as bad as 0.2 for 5 destinations. This is as much as 10 times lower than for the ER random graphs for certain problem sizes. So it appears that coocurrences on BA random graphs are quite easy to reconstruct, and it is likely that this is indicative of the performance on other scale free networks.

5.7 Random euclidean graphs

Euclidean graphs are graphs for which each node is associated with a point in space. Edges are either unweighted, or the edge weights correspond to distances in some metric. A random (euclidean) geometric graph is one for which each node is independently and uniformly distributed in some bounded region (such as the unit square), and nodes are joined by edges if and only if they are within some distance threshold d of each other. Random geometric graphs have been throughly studied, see Penrose [2003]. Random graphs tend to show very high clustering, as sets of nodes all within d of each other will form a clique, which is relatively likely for choices of d for which the graph is connected. Unlike all the other models considered here, they typically have large diameter and average path length. This is simply because nodes who are physically disparate in the underlying space have a lower bound on the number of

edges between them proportional to this disparity, and inversely proportional to the threshold parameter d.

For our tests we generated 50 node random geometric graphs within the unit square. Nodes were connected if they were within 0.2 of each other. This was high enough to ensure connectivity for the majority of generated graphs. Figure 5.12 shows the path length distribution that occurred. The large average path length that is typical of geometric graphs is evident. Some paths were as long as 16-18 nodes, due to the fat-tailed distribution. From the path lengths we would expect the reconstruction task to be harder, and indeed the symmetric error rate is much higher than for the other methods. At 10 destinations (50 coocurrences) the error rate is roughly 2.2, which is more than double the corresponding error with the other types of random graphs. The distribution of the error rate against the number of destinations is also distinctly different from that of the Erdős-Rényi and Watts-Strogatz models, as it doesn't show the interesting hump around 12 destinations, but rather decreases steadily as the number of destinations increases.

Along with the increased error rate is a large increase in the number of possible minimum-edge reconstructions. This is clear when merging of indistinguishable nodes is not performed. Several instances had more than 20 possible minimum-edge reconstructions! This would be due to one or more long paths which didn't overlap with other witnessed paths. Interestingly, the coverage is significantly lower than with any of the other random graph models. Even at 30 destinations (The majority of the graph's nodes), only 60% of the graphs edges were witnessed by coocurrences. This suggests that a large portion of edges are not needed for signal routing through such a network.

5.8 Tests on an Internet Topology graph

Perhaps the most prominent network of modern times, Internet routers and the fiber optic connections between them form a network. To determine how practical our BIP algorithm is, and to ascertain its performance on large networks, tests were conducted on the SKITTER network (Huffaker et al. [2005]). SKITTER is an Internet toplogy dataset; a subset of Internet routers and the connections between them, sampled over a long period of time so that the evolution of the Internet's topology is evident. SKITTER was sampled using the traceroute tool, which determines the set of routers (and there order) between a source computer and any destination. Several dozen machines were used to gather traceroute samples to over a million destinations. The network structure was later extracted from the SKITTER dataset and made available (University, Leskovec et al. [2005]).

The SKITTER network tested against has 1,696,415 nodes and 11,095,298 edges, and is one of the largest network structures published & freely available. For our tests 40 source nodes were randomly chosen, and shortest paths were plotted between those

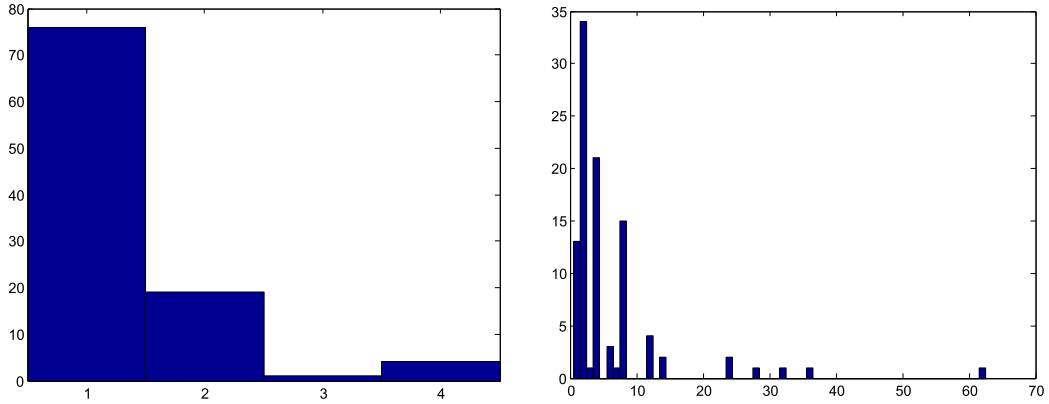


Figure 5.11: Number of solutions for each run for geometric random graphs. Runs without merging of indistinguishable node sets are on the right.

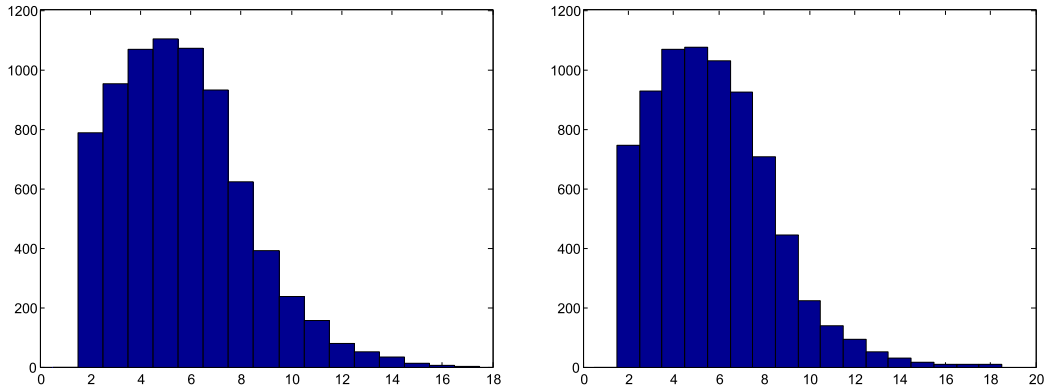


Figure 5.12: Path length for coocurrences generated on geometric random graphs

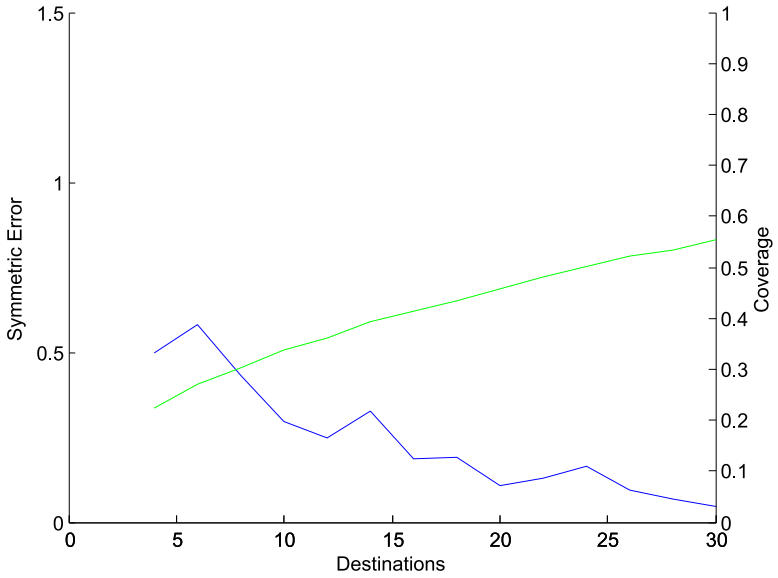


Figure 5.13: Coverage & error rate as number of coocurrence samples is increased for geometric random graphs

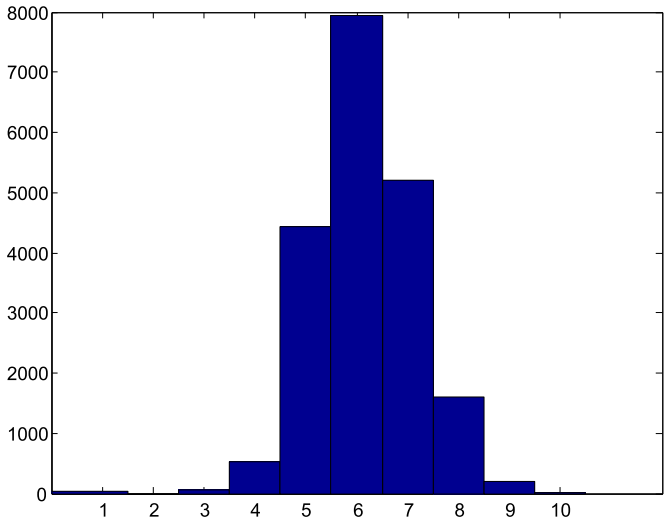


Figure 5.14: Path length for coocurrences generated on the SKITTER network

and 500 other randomly chosen nodes in the network. This resulted in 20,000 coocurrence samples, involving 8,272 nodes and 23,455 edges. Both the BIP method and the FM method were then used to reconstruct the paths using these samples. We were not able to test against the ML method, as our implementation could not handle such a large problem. The full BIP method was used including connectivity constraints.

Figure 5.14 shows the distribution of path lengths for the coocurrence samples. Despite the internet's nodes each having a geographic location, the distribution of path lengths does not resemble a random euclidean graph (see Figure 5.12). Instead, it resembles the other random graph models discussed, although with a median path length 2 greater, and average path length of 6.17. The small increase in average path length compared to the vastly smaller random graph models is typical of the $\ln \ln n$ average path length of some scale free networks. The average clustering coefficient of 0.296 is also typical of both scale free and small world networks.

Running the BIP solver took 83 minutes running on a single processor core on a desktop (non-professional) machine. The BIP constraints were generated in MATLAB, and took on the order of 3 hours to create. When constraint generation was used, it brought the total computation time down to 2 hours and 38m minutes. The 10 iterations of the FM algorithm took 75 minutes in total.

The preprocessor detailed previously was also used, resulting in 493 merge operations. The BIP method gave a reconstruction with symmetric error 868, compared to 2631 for the FM method. This is 3.0311 times less error when using the BIP method, resulting in only 3.7% wrong edges.

5.9 Tests on a Citation network

Citation networks are the most famous example of typically scale-free networks (de Solla Price [1965]). In order to determine the effectiveness of the BIP method on citation networks, a number of tests were performed in the arXiv HEP-PH citation graph dataset. First published as a dataset in the 2003 KDD Cup (**?**), this is a 34,546 node dataset consisting of all papers on high energy physics for which preprints were made available on arXiv, an e-prints hosting web site. An edge exists between two nodes if one there respective papers cites the other. Citations for papers outside of the arXiv database are not considered.

For our tests, 20 sources where chosen, and shortest paths were plotted between each of those sources and 500 other nodes in the network. These paths where then considered as coocurrences by dropping the order information. 11,693 edges were involved in these coocurrences, and the generated coocurrences had an average path length of 5.19. The citation network had an average clustering coefficient of 0.2962; virtually identical to the Internet topology graph considered in the previous section. The paths do not attempt to model any actual phenomena in citation networks, so this test can be considered as more of a general performance test than those on the Internet topology graph.

Total running time for the BIP method using constraint generation was 28 minutes, in which 0 constraints needed to be generated. Running time for 10 iterations of the FM method was 108 seconds. The preprocessing stages resulted in 290 node pairs being merged together, and the reconstruction error was found to be 518 for the FM method and 242 for the BIP method. This is an error of only 2% for the BIP method, resulting in 2.14 times lower error than the FM method.

Chapter 6

Tree Structured Coocurrences

In previous chapters we considered the problem of reconstructing coocurrence paths—signals that travelled from some source to a destination along a path in the network. In this chapter we consider a generalisation of this problem to tree structured coocurrences, where a signal starts at some source and travels through a tree within the network.

We propose a simplified version of the problem, where additional information is available about the relative time in which the signal passed through each node. The BIP method developed for path coocurrences is then adapted for this problem, and the resulting algorithm is found to be simpler and more efficient than the path coocurrence case.

6.1 Problem formulation

The natural generalisation of path coocurrences is to relax the requirement that they form a path, so that they are only required to be connected in the underlying network. It is natural to think of the signal that generated the coocurrence as traveling down a tree, as any set of connected nodes has a tree as a subgraph.

This problem is naturally harder, as there are more possible configurations of a set of nodes as a tree than as a path. Table 6.1 shows the number of possible reconstructions of n nodes under several different problem formulations, for varying n. Note that the number of undirected trees of n nodes corresponds to the well known Cayley's formula, $n^{(n-2)}$ (Cayley [1889], OEIS sequence A000272). This grows much faster than the number of undirected paths (which is just $n!$).

We also propose a related problem, which is perhaps closer to what might be encountered in applications.

Definition 6. ***Monotonic tree*** *coocurrences are tree coocurrences with additional side information for each coocurrence. Every node in a coocurrence X_i is assigned a 'time' within that coocurrence ($t_i : X_i \to \mathbb{N}$), subject to the following restrictions:*

- *The source (s_i) has time zero (i.e $t_i(s_i) = 0$).*

Table 6.1: Number of reconstructions for varying types of coocurrences

n	1	2	3	4	5	6	7
paths, known src & dest	*na*	1	1	2	6	24	120
paths, known src	1	1	2	6	24	120	720
Undirected tree	1	1	3	16	125	1296	16807
Monotonic tree	1	1	2	6	24	120	720

- *Each node's time is greater than or equal to that of it's parent in the tree.*

These times are local to that coocurrence, so that the same node in separate coocurrences can be assignened different times. This formulation is designed to correspond to the case where it is possible to determine at what time a signal reaches a node. If we were considering path based coocurrences, this would be make the problem trivial, as the timing information would completely determine the path. In the case of trees however, it is a challenging problem. Table 6.1 shows the number of possible reconstructions of a coocurrence of n nodes given this time information, for varying n. It scales exactly the same as the case of paths with known source, so is likely to be a far more tractable problem than the undirected tree case.

Theorem 3. *Let $m(n)$ denote the number of (labeled) monotonic trees for a given n. Then $m(n) = (n-1)!$.*

Proof. We proceed by induction. For n=1, clearly only one configuration exists, so $m(1) = 1 = 0!$. Suppose that $m(n-1) = (n-2)!$. We will consider each of these $(n-2)!$ tree configurations of $n-1$ nodes, and determine how many ways an additional nth node can be added to yield monotonic trees. Without loss of generality, suppose that the time for node n is greater or equal to each existing node (the nodes can be relabeled such that this is true). Then for any monotonic configuration of the $n-1$ nodes, it is possible to add node n as a child of any node in that tree. As there are $n-1$ nodes, this means that there are $n-1$ configurations of the n nodes for each of the configurations of $n-1$ node set. Therefore the total number of monotonic configurations is $m(n) = (n-2)! \cdot (n-1) = (n-1)!$ □

So we consider two reconstruction problems in this section. The first is the reconstruction of a graph when given a set of tree coocurrences, and the second case is when we are given monotonic tree coocurrences. As noted above, in the monotonic tree case we are given additional 'timing' information for each coocurrence, which greatly reduces the total size of the solution space of the problem, although it is still exponential in the path lengths, just like the path coocurrence case.

6.2 Potential applications

Tree coocurrences are better models of the sort of communication that occurs on social networks. When a bit of information travels through a community, often once a person has that information, they communicate it to several others. So modelling the communication of that information as a signal, it forms a tree structure as opposed to a path. 'Sampling' the signal by asking people if they knew that particular bit of information would form a coocurrence, so a set of such samples is an instance of the tree coocurrence problem. This idea translates into the digital age in the form of email communications that get forwarded between many people. If those emails contained a link then a similar communication could be performed based on who clicked the link. This problem is even an example of monotonic tree coocurrences, if the time at which the links are clicked is recorded as well. Such a reconstruction would be error prone; there is no guarantee that each person who forwarded the email clicked the link, and so false edges could be introduced due to this.

Another example of the tree coocurrence problem occurs in online social networks where people can declare their membership of groups. By monitoring and recording who joins each group, and when, a set of monotonic tree coocurrence samples is formed. This requires the assumption that people find out about the group from their friends; people who are not friends of an existing member who join the group will cause false edges to appear in the reconstruction. Depending on the specifics of the online social network, this may not occur often.

6.3 BIP reconstruction

The method described here is inspired by the weak BIP path coocurrence reconstruction algorithm. The main purpose of this section is to give a baseline algorithm, determine if reconstruction is practical, and so indicate whether further research on algorithms for this problem is worthwhile. We focus on the monotonic tree coocurrence case.

As we did for path coocurrences, we denote each of k coocurrences as $X_{1\ldots k}$, and the source of each as s_i. For consistency with the path case we consider the source s_i as not being part of X_i. Unlike the path case, it is more convenient to reconstruct the directed graph structure. When we reconstructed the undirected structure for path coocurrences, this could be converted into a directed structure by tracing the paths through the graph, and orientating edges accordingly. This is not necessary for this tree reconstruction algorithm.

Let x be a binary vector representing the set of (directed) edges, so that $x_{i\to j} = 1$ if edge (i, j) exists, and 0 otherwise. We will formulate a set of constraints, that together ensure that any x that satisfies the constraints is consistent with the tree coocurrences $X_{1\ldots k}$.

The constraints corresponding to coocurrence X_i are as follows. Firstly we define a subset E_i of the set of possible edges E, that could occur in a possible reconstruction of coocurrence i when other coocurrences are ignored. Making use of the timing information, this gives:

$$E_i = \{(u,v) \mid t_i(u) \leq t_i(v), \text{ and } u,v \in X_i \cup \{s_i\}\}$$

E_i is just the set of edges that don't violate the monotonicity condition. Given this definition, we define the following constraints:

Size constraint

$$\sum_{(u,v)\in E_i} x_{u\to v} \geq |X_i|$$

Weak degree constraints

$$\forall v \in X_i, \sum_{u\in\{s_i\}\cup X_i, \text{ and } (u,v)\in E_i} x_{u\to v} \geq 1$$

Only the weak degree constraints are necessary to ensure that the coocurrence forms a tree structure. The other set of constraints help guide the integer programming solver towards a solution quicker.

Unlike the path coocurrence case, the degree constraints alone completely enforce the coocurrence structure. Implementation is significantly simpler, and fewer constraints results in the BIP solver computing the solution quicker.

6.4 Synthetic experiment results

Tests were conducted on 50 node Erdos-Renyi graphs, similar to the tests performed in the path coocurrence case in Chapter 4. Each coocurrence was generated by first sampling *i.i.d.* from the set of nodes to determine the sources. Starting at each source, a coocurrence was generated by keeping track of a 'fringe' queue. Each neighbour of the source was added to the queue with probability r. Each node in the queue was processed in a similar way, with their neighbours added to the queue with probability $r \cdot b^d$ where d is the depth of the node being processed (The source was defined as depth 0). This method produces a geometric falloff, and the parameters b & d can be adjusted to effect tree size.

For Figure 6.1, the parameters were set at $r = 0.25$ & $b = 0.5$, and the number of coocurrences was varied between 100 and 600. The graph density was set at 4, and otherwise test methadology was identical to the tests in Chapter 4. The distribution of the size of the coocurrences is shown in Figure 6.2.

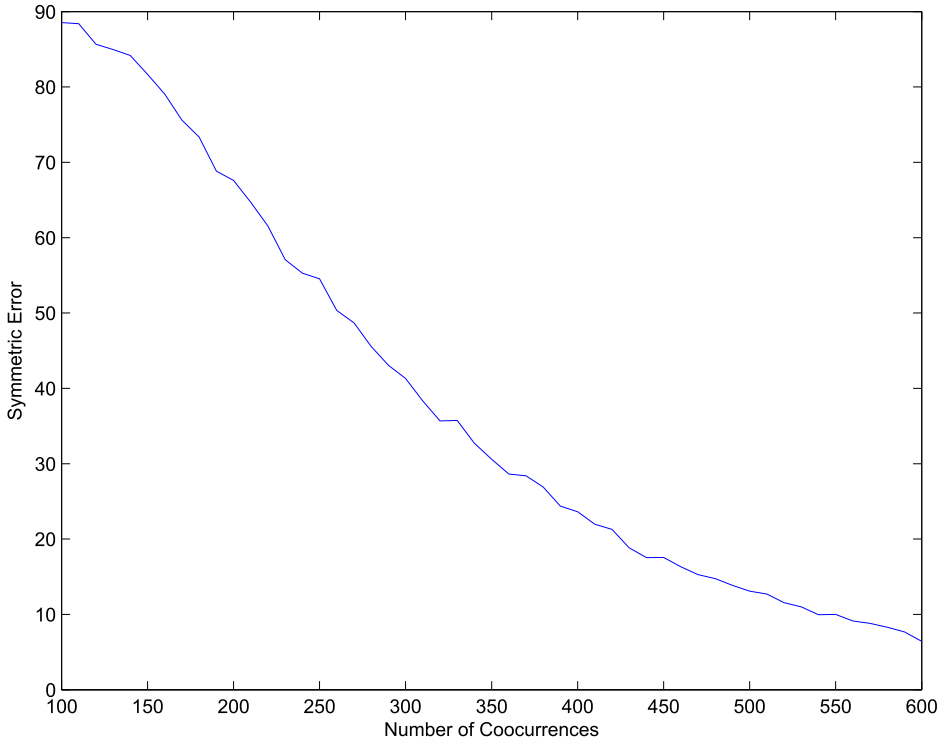


Figure 6.1: Error rate for tree coocurrence reconstruction

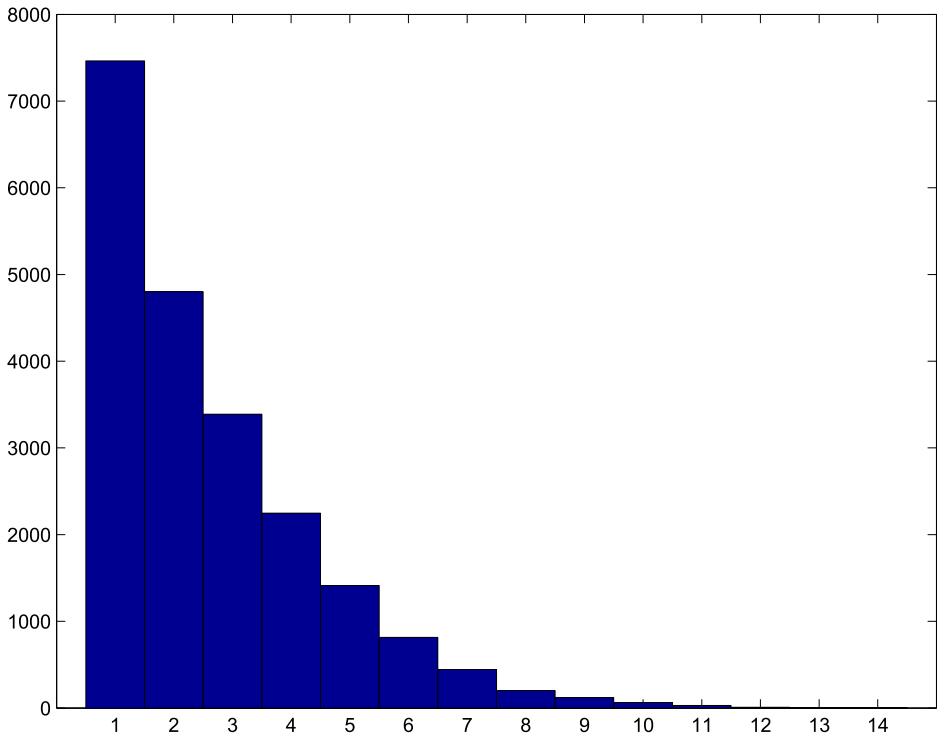


Figure 6.2: Distribution of the number of nodes in the generated coocurrences

The error rate is significantly higher than what occurs with path structured coocurrences. However even with the error rate of 60 for 100 coocurrence samples, the reconstructed network resembles the true reconstruction enough for it to be useful. For a very large number of samples, the error rate drops significantly, in practice the error rate of 15 at 600 samples is only an average of 7 misplaced edges. Note that the coocurrence size distribution results in a large portion of 1 node coocurrences, which don't contribute to the reconstruction. Considering this, on the whole networks can be usefully reconstructed from monotonic tree coocurrence samples.

Chapter 7

Conclusion

In this work we developed a novel method for solving the network reconstruction from coocurrence problem. We defined the *shortcut free* property for routing schemes, a weakened form of the deterministic shortest path property. Under the assumption that a set of coocurrences used a deterministic shortcut free routing scheme, we showed that the correct reconstruction has the minimal number of edges possible for a reconstruction. We also showed that all minimal edge reconstructions of a given set of coocurrences are shortcut free, so that with utilizing other information about the network structure any minimal edge reconstruction was as good as any other.

The shortcut free property appears to be the correct middle ground between a local routing assumption (i.e. random walk) and a global assumption such as shortest path routing. It can be thought of as path local, although the global minimal edge reconstruction property implies it. The minimal edge reconstruction is the intuitively correct solution for small problems when they are presented to people. The fact that deterministic shortcut free routing implies that it is the best reconstruction method is further evidence that it is the correct routing assumption for solving many real world problems.

By assuming the that shortcut a set of coccurrences is shortcut free, we showed that it was possible to encode the network reconstruction problem as a set of linear inequalities, and using a binary integer programming solver it was possible to find the minimal edge reconstruction. This method, while potentially having exponential worst case performance, was found to be efficient in practice, allowing the solving of problems with tens of thousands of concurrences and millions of nodes. It also had a number of interesting properties; it was possible to recover *all* minimal edge reconstructions, and a weakened version of the constraints could be used when the coocurrences were not shortcut free.

In comparisons against existing methods on synthetic problems, this new method consistently out-performed the existing state of the art method for all but trivially sized problems, with margins of 14-57% less error. Tests were also performed on several real world networks, including a 1.6 million node internet topology network. Twenty thousand coocurrences were generated, and a reconstruction was performed that took on the order of $2\frac{2}{3}$ hours. The symmetric error between the true path structures and

the reconstruction was only 3.7%. Error rates were between 2 and 3 times lower then with the FM method.

A new problem was also considered, that of tree structured coocurrences. This has wide applicability to social networks, and any other network where signals do not follow a path. Tree structured signals form a middle ground between point to point and multi-cast models. A variant of the optimization algorithm proposed for path coocurrences is given. It turns out that encoding tree structured coocurrences is efficient when additional signal timing information is available, a case that we denote as monotonic tree coccurences. Then the set of constraints is simpler, and reconstruction is generally faster then the path case. However, reconstruction error is much higher then the path case, ranging from 70 to 10 errors on average, for between 100 and 600 coocurrence samples on a 50 node network.

List of Figures

Bibliography

Egon Balas and Robert Jeroslow. Canonical cuts on the unit hypercube. *SIAM Journal on Applied Mathematics*, 23(1):61–69, 1972. doi: 10.1137/0123007. URL http://link.aip.org/link/?SMM/23/61/1.

Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

Albert-Laszlo Barabasi and Reka Albert. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–94, 2002.

Michel Berkelaar, Kjell Eikland, and Peter Notebaert. lp_solve v5.5.2. 010. Open source (Mixed-Integer) Linear Programming system, Multi-platform, pure ANSI C / POSIX source code, Lex/Yacc based parsing.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Bela Bollobas. *Random Graphs*. Academic Press, INC, 1985.

Rui Castro, Mark Coates, Gang Liang, Robert Nowak, and Bin Yu. Network tomography: recent developments. *Statistical Science*, 19:499–517, 2004.

Arthur Cayley. A theorem on trees. *Quarterly Journal of Pure and Applied Mathematics*, XXIII:376–378, 1889.

Reuven Cohen and Shlomo Havlin. Scale free networks are ultra-small. *Physical Review Letters*, 90:5, 2003.

Derek J. de Solla Price. Networks of scientific papers. *Science*, 149:510–515, 1965.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

Bradley Huffaker, Young Hyun, Dan Andersen, and kc claffy. The skitter as links dataset. http://www.caida.org/data/active/skitter_aslinks_dataset.xml, 2005.

Marcus Hutter. *Universal Artificial Intelligence*. Springer, 2005.

Donald E. Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. Addison-Wesley, 1997.

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

George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Publication, 1988.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 2nd edition, 1998.

Mathew Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

Michael G. Rabbat, John R. Treichler, Sally L. Wood, and Michael G. Larimore. Understanding the topology of a telephone network via internally-sensed network tomography. In *Procedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 977–980, Philadelphia, PA, March 2005.

Michael G. Rabbat, Mrio A. T. Figueiredo, and Robert D. Nowak. Network inference from co-occurrences. Technical report, 2006.

Stanford University. Stanford large network dataset collection. http://snap.stanford.edu/data/index.html.

Y. Vardi. Network tomograph: Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91:365–377, 1996.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.