

Belief Propagation convergence in networks:

April 6, 2009

Lisa Farnell, Jessica Havelock, and Claudette Yungwirth

Supervisor: Professor Serdar Yüksel

1 ABSTRACT

The belief propagation algorithm is an algorithm which attempts to draw conclusions from a network based off of limited information. In this project, we examine the algorithm in detail, analyzing its behaviour as certain input parameters are varied. Particularly, we are interested in the algorithm's convergence behaviours. This project aims to answer the questions: "When will the belief propagation algorithm converge to yield inferred results?" and "When will these inferred results be correct?". The properties of the belief propagation algorithm are not fully understood in loopy systems. Our research allows the use of the algorithm to be extended and applied to certain loopy networks. This is accomplished by understanding the algorithm's specific convergence properties.

First we analyzed an area of the algorithm that is currently well understood in order to gain comprehension of its processes. This consisted of analyzing the algorithm's performance under Markov Chain and Tree-Structured networks as other parameters are altered while the network structures remain constant. MATLAB simulations were then created to confirm convergence as found in initial research.

Secondly, our analysis was extended to loopy networks, meaning those containing cycles. The algorithm's behaviour in loopy networks is quite different. In order to understand when the algorithm will converge, we've developed a procedure which can be applied to all loopy networks, yielding areas where the belief propagation algorithm will converge locally. We will also identify points to which the algorithm can converge for a number of situations. The results allow the belief propagation algorithm to be used in certain loopy networks with a greater certainty of accurate results.

2 ACKNOWLEDGEMENTS

We would like to thank Dr. Serdar Yüksel for his guidance, time and support throughout this project.

Contents

1	ABSTRACT	1
2	ACKNOWLEDGEMENTS	2
3	INTRODUCTION	6
3.1	Background Information	6
3.2	Objective	8
3.3	Initial Research	8
3.4	Defining Variables	9
4	DESCRIPTION, THEORY, AND APPROACH	9
4.1	Message Passing in Markov Chains, Tree structures, and Loops	9
4.2	Analysis of Belief Convergence	14
5	DESIGN	14
5.1	Design Goal	14
5.2	Design: Procedure for Showing Local Convergence	15
5.2.1	Constructing the Update Function T	15
5.2.2	Examining Fixed Points of T and Contractions	18
5.2.3	Fixed Point Argument	19
5.2.4	Contraction Argument	20
5.2.5	Linearizing	21
5.2.6	Norm Discussion	21
6	RESULTS	22
6.1	Convergence of Markov and Tree-Structured Networks in MAT- LAB Simulation	22
6.2	Convergence in a Fully Connected Network	23
6.2.1	Determining Fixed Points	23
6.2.2	Showing Convergence To Fixed Points	24
6.3	Extending Our Analysis	26
7	DISCUSSION	27
8	CONCLUSION	28
8.1	Concluding Remarks	28
8.2	Future Work	29

A	APPENDIX - Proofs, Derivations and Long Equations	31
A.1	Markov Derivation	31
A.2	Tree-Structure Derivation	31
B	APPENDIX - Maple code and Results	33
B.1	Maple Code for the Three-Node Fully-Connected Loop	33
B.2	Maple Code for the Five-Node Fully-Connected Loop	40
B.3	Summary of Fixed Points from Maple Results	47
C	APPENDIX - MATLAB Simulations and Maple analysis Code	53
C.1	Markov network code	53
	C.1.1 Markov network code	53
	C.1.2 Tree-Structured network code	56
C.2	Fully Connected graph network code	59

List of Figures

1	Interconnected network	9
2	Markov interconnected network	10
3	Tree structured graph featuring both observable and unobservable nodes	11
4	Tree structured graph used for analysis	11
5	Simple loopy graph	12
6	Illustration of message passing considered when analyzing different nodes of a loopy network	13
7	Fully-connected five-node network	17
8	Local contraction around a fixed point B^* . Specifically, this is the effect of the update function $T(x)$ in a small neighbourhood around B^*	19
9	Tree node Markov Chain MATLAB simulation output	23

3 INTRODUCTION

3.1 Background Information

Belief propagation algorithms are currently used in order to analyze systems which are modelled by networks. Whenever graph theory can be used to model information with nodes and connecting communications channels, the belief propagation algorithm can be used to infer information. This is most applicable when the information of the system - or nodal beliefs of the network - are limited or unclear, possibly due to noise. Such systems can be modelled by networks that include both nodes with observable and with unobservable beliefs.

The belief propagation algorithm can be used in a vast range of network modelling areas. When analysis is needed for systems with limited or noisy information, the belief propagation algorithm can be used to infer needed information or to draw conclusions based on initial observed beliefs. Models used in sensor networking, weather predictions, gossip algorithms, error-correcting codes, speech recognition, and many other areas make use of the belief propagation algorithm [1][2][3].

A system that is modelled in a graph will consist of several nodes - which can have observable or unobservable beliefs - and connecting lines or "paths". During the belief propagation algorithm, observed nodes only transmit their information, and do not receive information from neighbouring nodes to update their belief [4]. Each unobserved node is updated at each iteration based on the information received from neighbours [5].

In graphical representation of a network, a node i is a neighbour to a node j if there is a line connecting them, implying belief information can be passed along the path from one node to the other [6]. Connecting lines between nodes therefore indicate conditional dependence between nodes [2][1]. For a given node in a network, any node that passes information to it is considered its parent node, while any node to which the given node passes its information to is considered its child node.

Networks can have directed or undirected paths. In directed paths, information can only be passed in the indicated direction. For example, if node i

and node j are connected with a path directed from i to j , then it is implied that node j can receive information from node i , but node i cannot receive information from node j [2]. Graphs studied within in the scope of this project are primarily undirected, implying that communication can occur in both directions on any given path.

In modelled networks, each node has its own initial belief at some time, t . Given the structure of a network, each node can share this initial belief with its neighbouring nodes during the first iteration of the algorithm. Each node will then update its own belief based on its received information, and will pass this updated belief to its neighbours during the next iteration, at time $t + 1$ [1][7][8][9]. Since each node may only share data with its neighbours, information from one node may need to pass through several others before its information is received by a non-neighbouring node. The message passing works in iterative steps, until each node's belief remains constant at successive iterations [1][7][9].

As information is being passed during an iteration of the algorithm, a node will receive the beliefs from each of its neighbours. Its belief is then updated. This updated belief is then passed to its neighbours in the next iteration, while at the same time new information is received from each of its neighbours which is used to update its belief once again. Thus, the belief propagation algorithm is a recursive process which continues until a node's belief remains constant after multiple successive iterations.

When beliefs of all nodes remain constant after a number of algorithm iterations, the beliefs are said to have converged[1]. The idea of convergence in networks causes certain questions to arise as the belief propagation algorithm is implemented, such as: "Do beliefs always converge?" and "Which network structures will allow for convergence?". It is important to note that each node has a probability distribution describing how much it trusts its own opinion, and how much it trusts the messages received from each of its neighbours. These probability distributions are utilized by a node to update its personal belief upon the receipt of information from its neighbours.

The algorithm is currently known to converge in linear networks such as Markov chains and tree structured networks [7], although convergence in net-

works with cyclic structures remains unclear. Background research indicates that in networks involving cycles, the algorithm does not necessarily converge [5]. Sometimes the algorithm causes the belief of a node to be an oscillation between two beliefs at successive iterations [10][4]. In other instances, the algorithm has been seen to converge to an answer that is incorrect, while in others convergence does not occur at all [4].

3.2 Objective

Our objective is to analyze the belief propagation algorithm in detail in order to understand its behaviour. To be more specific, our goal is to determine the structure of networks for which the algorithm will converge. This consists of analyzing how convergence of the algorithm behaves as network structures and probability distributions of variables are varied. The algorithm's behaviour in networks containing cyclic paths is not fully understood in belief propagation. Due to this fact, loopy belief will be the focus this project.

3.3 Initial Research

The first task of this project was to understand the belief propagation algorithm. Dr. Yüksel provided our team with some basic examples of applications, and some background information about the algorithm and interconnected networks to which it can be applied. Networks to which the algorithm can be applied vary in complexity and structure. Figure 1, seen below, is an example of a complex interconnected network.

Networks are constructed of nodes and interconnecting lines. In belief networks, each node has a certain belief and the lines between neighbouring nodes represent inter-nodal communication. Initially, unobservable nodes base beliefs off the values of observable nodes. As the algorithm progresses, beliefs are communicated between nodes, and updated beliefs are generated. Convergence occurs when beliefs stay constant as updates continue to occur.

Preliminary research was conducted in order to understand the main ideas and applications behind the belief propagation algorithm. The initial general relationship which is further analyzed in great detail can be seen below in Equation 1.

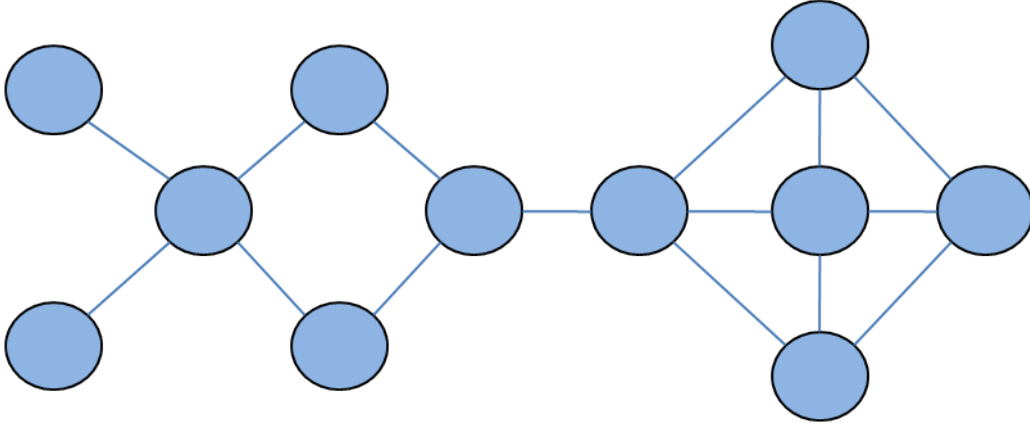


Figure 1: Interconnected network

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{Parents}(X_i)) \tag{1}$$

3.4 Defining Variables

In this paper, we have defined our variables as follows:

$P(X_i)$ - the unobserved, or inferred, belief of node i

$P(Y_i)$ - the observed belief of node i

$b_i = P(X_i | Y_1 Y_2 \dots Y_n)$ - the belief of node i for a network of n nodes

4 DESCRIPTION, THEORY, AND APPROACH

4.1 Message Passing in Markov Chains, Tree structures, and Loops

Our research in the belief propagation algorithm started with small networks with specific structures. Markov chains, a simple type of interconnected graph, were the first type to be studied. A Markov chain of length

3 with probabilistic relationships $P(X_1X_2X_3Y_1Y_2Y_3) = P(Y_1|X_1)P(Y_2|X_2)P(Y_3|X_3)P(X_2|X_1)P(X_3|X_2)$ is illustrated below in Figure 2. The nodes in Figure 2 that are labelled with X_i are unobservable nodes, while those labelled Y_i have observed beliefs.

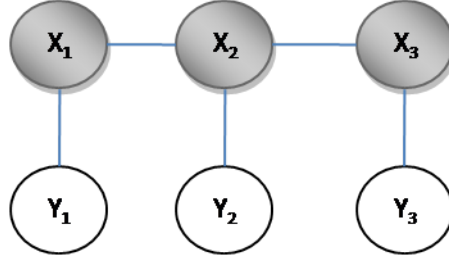


Figure 2: Markov interconnected network

Our team analyzed these Markov chain networks and was able to successfully derive equations to calculate probability of unobserved node values given the observable values. Shown below is the derived equation for the value of X_i given the observed Y_j nodes for $j = 1, 2, \dots, n$ for an n -node Markov network. These probabilistic values for X_i are described as function of neighbouring nodes' messages at the previous time step. The derivation for this result for the 3-node Markov chain can be seen in Appendix A.1.

$$P(X_i|Y_1\dots Y_n) = \frac{\sum_{X_{i-1}} [P(Y_i|X_i)P(X_i|X_{i-1})P(X_{i-1}|Y_{i-1}Y_{i-2}\dots Y_1)]}{\sum_{X_i} \left(\sum_{X_{i-1}} [P(Y_i|X_i)P(X_i|X_{i-1})P(X_{i-1}|Y_{i-1}Y_{i-2}\dots Y_1)] \right)}$$

for $i = 1\dots n$

(2)

For the three-node Markov Chain, the equations shown above are used to find the probabilistic updated beliefs of the X_2 and X_3 nodes from a forward sweep of information ($X_1 \rightarrow X_2 \rightarrow X_3$), and can be used with reversed subscripts to express the probability of belief values of the X_2 and X_1 nodes for a backward sweep ($X_3 \rightarrow X_2 \rightarrow X_1$).

More complex tree structured graphs were also examined during the early stages of the project. A tree structured graph is shown below in Figure 3.

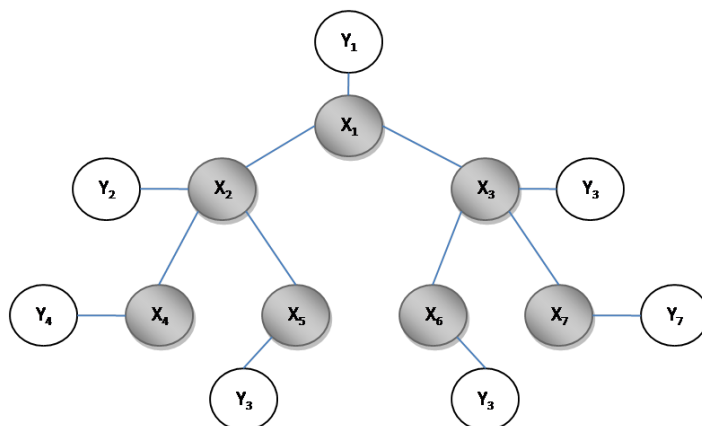


Figure 3: Tree structured graph featuring both observable and unobservable nodes

As with the notation used for the Markov chain which is illustrated in Figure 2, nodes labelled X_i are unobserved nodes, while Y_i nodes are observable. For the tree structure, information from the "bottom" nodes is passed up the tree to the "top" node in an upward sweep of information. After beliefs have been updated from this upward sweep, new information is passed back down to the "bottom" nodes in a downward sweep of information. We began our analysis of tree-structured networks by analyzing a simple three-node tree structure with probabilistic relationships $P(X_1 X_2 X_3 Y_1 Y_2 Y_3) = P(Y_1 | X_1) P(Y_2 | X_2) P(Y_3 | X_3) P(X_1 | X_2 X_3) P(X_3) P(X_2)$ which is seen below in Figure 4.

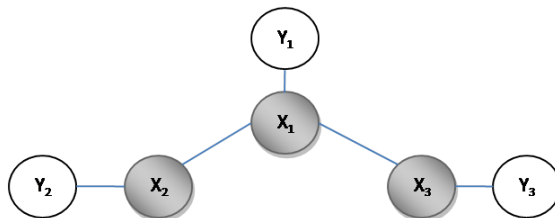


Figure 4: Tree structured graph used for analysis

The downward sweep of information in the tree structured diagram involves similar message passing as in a Markov chain. As a result, equations

expressing the probabilistic beliefs of the X_i nodes as information is passed in a downward sweep through the "branches" of the tree have the same form as the equations shown above for Markov chains.

An expression for an upwards sweep of a general tree-structured graph was also derived and can be seen below in Equation 3. This method of derivation allows for tree structured networks with n nodes to be analyzed. Note that nodes that are "above" the subject node are considered its parents, while nodes that are "below" it are considered its children. The derivation for this equation can be seen in Appendix A.2.

$$P(X_i|Y_1..Y_n) = \frac{P(X_i|Y_i) \prod_{c \in \text{children } X_i} \left[\sum_{X_j} P(X_j) P(X_j|Y_j) P(X_i|X_j) \right]}{[P(X_i)]^{\# \text{ children } X_i} \sum_{X_i} \frac{P(X_i|Y_i)}{[P(X_i)]^{\# \text{ children } X_i}} \left[\prod_{X_j \in \text{children } X_i} \sum_{X_j} P(X_i|X_j) P(X_j|Y_j) P(X_j) \right]}$$

where c =children

(3)

The next type of graph that was investigated was a loopy network. A loopy network is characterized by a cycle in the graph structure[1][2][10][11]. Figure 5, seen below shows an example of a three node loopy network.

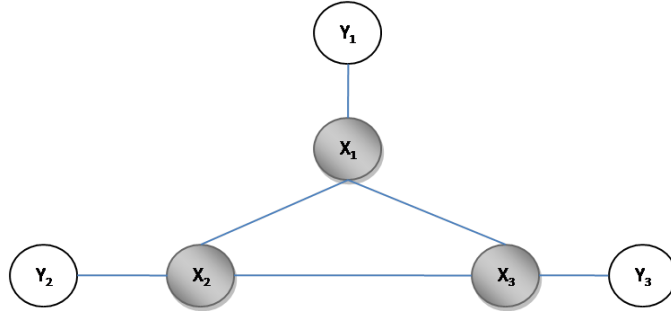


Figure 5: Simple loopy graph

When considering an individual node in a loopy network, the equations used to express the nodes' updated belief based on its neighbours' information

are the same as those used for tree structured graphs[5]. Figure 6 seen below shows an illustration of this for the three node loopy network. Relations between certain nodes can be disregarded, as depicted, when the passing of information between these nodes is irrelevant to the node that is the subject of the specific equation.[7][9]

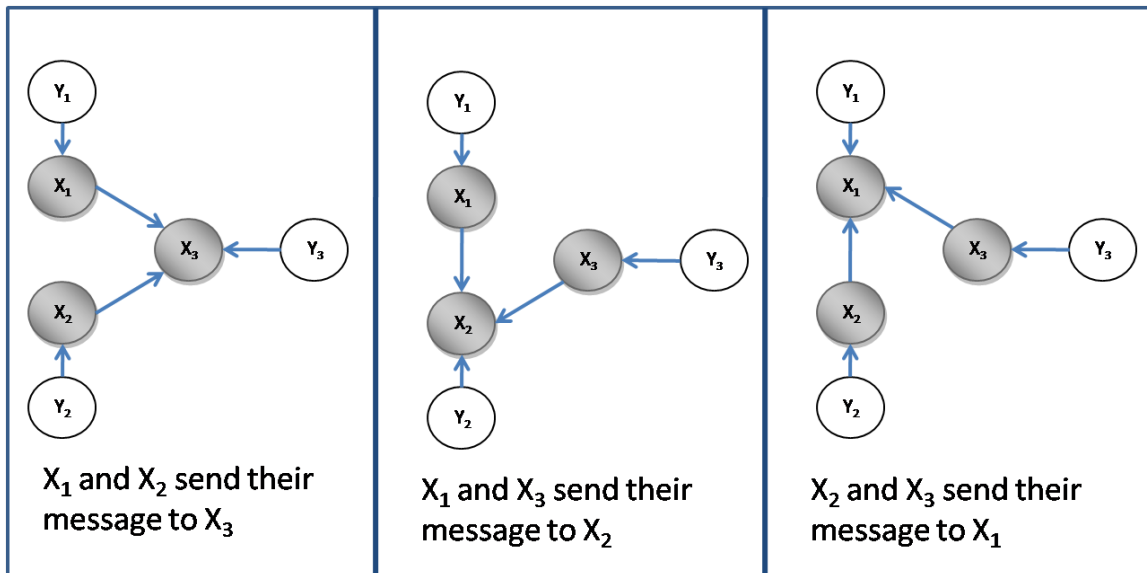


Figure 6: Illustration of message passing considered when analyzing different nodes of a loopy network

After an equation was individually derived for each node of a loopy network, a vector function of these equations was built. This vector equation was an update function for the entire network that expressed the beliefs of each unobservable node as a probabilistic function of the messages passed from its connected neighbours at the previous time step. We used the update equation, named T , in this vector form to simplify the analysis of the trends in updating beliefs. Update equations T of specific networks that were analyzed can be seen in Section 5.2.1.

To demonstrate the mathematical results of the update equation analysis, simulations of the belief propagation algorithm for these three specific graph

structures were created using MATLAB. A description of these simulations and their results can be found in Section 6.

4.2 Analysis of Belief Convergence

Graphs with Markov chain or tree structures have been previously shown to converge under the belief propagation algorithm [8][7], and can intuitively be realized when considering how information is passed within these graphs. In both of these structures, sweeps of information in opposing directions will cause information from all nodes to be passed to every other node, and allow for belief updates to occur with a complete set of neighbouring information. For both of these graph structures, the probabilistic beliefs of the unobservable nodes remain constant after a finite number of algorithm iterations.

Loopy networks have a more complicated structure than Markov chains and tree networks; the issue of double-counting arises. Double counting occurs when a node receives information from another node from two different paths. In the case of the network depicted in Figure 5 above, node X_1 will receive information from X_3 that is passed directly. Node X_1 will also receive information from X_2 that has previously been influenced by X_3 's information. In this way, the information from X_3 will be sent to X_1 via two different paths.

The effects of double counting on convergence were investigated by considering the update function, T , of a loopy network. A fixed point was found mathematically by solving the vector equation shown below. T was then linearized around this fixed point and analyzed for beliefs in a small neighbourhood around the fixed point. The creation and analysis of the update function further is discussed in Sections 5.2.1 and 6.2.2.

$$(B_t) = T(B_{t-1})$$

5 DESIGN

5.1 Design Goal

Our goal for our design is to construct a procedure to determine when a simple loopy network will converge to a belief under specific conditions.

These conditions will possibly restrict the following: communication between nodes, how much one node's belief depends on another's belief, ie $P(X_1|X_2)$, the accuracy of initial conditions, the distribution of the beliefs and possible others.

5.2 Design: Procedure for Showing Local Convergence

5.2.1 Constructing the Update Function T

Our designed procedure begins with the construction of a vector update function which we call T. In order to analyze the methodology behind the construction of T, we first looked at a specific loopy network. This network is the simple three-node loop depicted in Figure 5. The transformation function, T, for this loopy network is seen below in Equation 4.

$$\begin{aligned}
 & T(B) = \\
 & \left[\begin{array}{l}
 \frac{b_1^1 \left[\sum_{i=0}^1 P(Y_2) b_2^i P(X_1|X_2=i) \right] \left[\sum_{j=0}^1 P(Y_3) b_3^j P(X_1|X_3=j) \right]}{[P(X_1)]^2 \sum_{k=0}^1 \frac{b_1^k}{[P(X_1=k)]^2} \left[\sum_{l=0}^1 P(X_1=k|X_2=l) b_2^l P(X_2=l) \right] \left[\sum_{m=0}^1 P(X_1=k|X_3=m) b_3^m P(X_3=m) \right]} \\
 \frac{b_2^1 \left[\sum_{i=0}^1 P(Y_1) b_1^i P(X_2|X_1=i) \right] \left[\sum_{j=0}^1 P(Y_3) b_3^j P(X_2|X_3=j) \right]}{[P(X_2)]^2 \sum_{k=0}^1 \frac{b_2^k}{[P(X_2=k)]^2} \left[\sum_{l=0}^1 P(X_2=k|X_1=l) b_1^l P(X_1=l) \right] \left[\sum_{m=0}^1 P(X_2=k|X_3=m) b_3^m P(X_3=m) \right]} \\
 \frac{b_3^1 \left[\sum_{i=0}^1 P(Y_2) b_2^i P(X_3|X_2=i) \right] \left[\sum_{j=0}^1 P(Y_1) b_1^j P(X_3|X_1=j) \right]}{[P(X_3)]^2 \sum_{k=0}^1 \frac{b_3^k}{[P(X_3=k)]^2} \left[\sum_{l=0}^1 P(X_3=k|X_2=l) b_2^l P(X_2=l) \right] \left[\sum_{m=0}^1 P(X_3=k|X_1=m) b_1^m P(X_1=m) \right]}
 \end{array} \right]
 \end{aligned}
 \tag{4}$$

This function was derived from the update equations for a tree structured graph that was determined in the first half of the project. Due to the nature of message passing involving only neighbouring nodes, the conditional probability $P(X_1|Y_1, Y_2, Y_3)$ found for the tree-structured network seen in Figure 4 is also the first entry of the update function for the three-node loopy graph seen in Figure 5. This reasoning is also illustrated in Figure 6 as described earlier. The transformation function T will be the updating function for all unobserved nodes at time t , where its input parameters will be the beliefs at

time $t - 1$. The equation from which the T-vector was created is of the form $B_t = TB_{t-1}$, as shown in Equation 5.

$$\begin{bmatrix} b_{1t} \\ b_{2t} \\ b_{3t} \end{bmatrix} = T \begin{bmatrix} b_{1t-1} \\ b_{2t-1} \\ b_{3t-1} \end{bmatrix} \tag{5}$$

Where,
 B_t = the beliefs of unobserved nodes at time t
 B_{t-1} = the beliefs of unobserved nodes at time $t - 1$
 T = generated transformation

Since B_i is a probability distribution of beliefs, for $b_i \in B_t$, we know that $b_i \in [0, 1]$ for $i = 1, 2, 3$. This means that for $B_t \in R^3$, $B_t = [b_1, b_2, b_3]^T$. More generally for a loop of size n , $b_i \in [0, 1]$ $i = 1, 2, \dots, n$ and $b_t \in R^n$ with

$$B_t = \begin{bmatrix} b_1 \\ | \\ b_n \end{bmatrix} \tag{6}$$

The updating function T is updating the belief at nodes X_1 , X_2 , and X_3 based off of each X_i 's own belief and the belief of its neighbours at time $t - 1$. This is done simultaneously for each node in the network. This simultaneous passing of information is illustrated in Figure 6.

In a similar manner, probabilities can be found for any structure of network in order to produce the network's update function T. This analysis was extended to the fully-connected five-node network which is illustrated below in Figure 7.

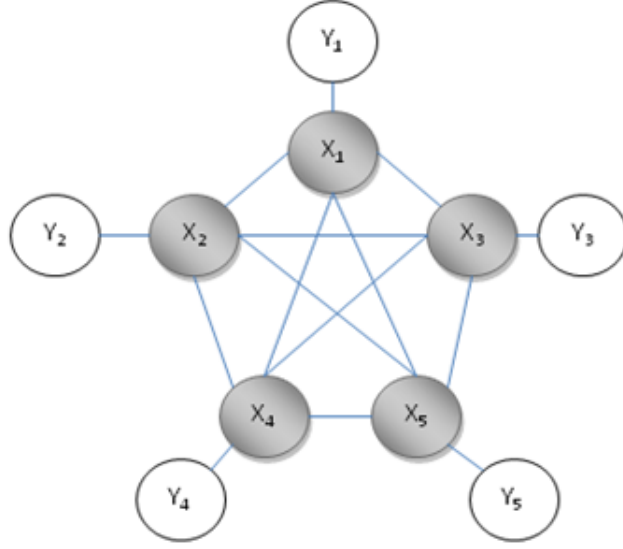


Figure 7: Fully-connected five-node network

The generated conditional probability for the network in Figure 7 for $P(X_1|Y_1Y_2Y_3Y_4Y_5)$ can be seen below:

$$P(X_1|Y_1, \dots, Y_5) = \frac{P(X_1|Y_1) \prod_{i=2}^5 \left[\sum_{X_i=0}^1 P(X_i|Y_i) P(Y_i) P(X_1|X_i) \right]}{P(X_1)^2 \sum_{X_1=0}^1 \left(\frac{P(X_1|Y_1)}{P(X_1)^2} \prod_{i=2}^5 \left[\sum_{X_i=0}^1 \frac{P(X_i|Y_i) P(Y_i) P(X_1|X_i)}{P(X_i)} \right] \right)} \quad (7)$$

The generated update function for the five-node fully-connected network seen in Figure 7 is shown below.

$$T(B) = \left[\begin{array}{c} \frac{P(X_1|Y_1) \prod_{i=2,3,4,5} \left[\sum_{X_i=0}^1 P(X_i|Y_i)P(Y_i)P(X_1|X_i) \right]}{P(X_1)^2 \sum_{X_1=0}^1 \left(\frac{P(X_1|Y_1)}{P(X_1)^2} \prod_{i=2,3,4,5} \left[\sum_{X_i=0}^1 \frac{P(X_i|Y_i)P(Y_i)P(X_1|X_i)}{P(X_i)} \right] \right)} \\ \frac{P(X_2|Y_2) \prod_{i=1,3,4,5} \left[\sum_{X_i=0}^1 P(X_i|Y_i)P(Y_i)P(X_2|X_i) \right]}{P(X_2)^2 \sum_{X_2=0}^1 \left(\frac{P(X_2|Y_2)}{P(X_2)^2} \prod_{i=1,3,4,5} \left[\sum_{X_i=0}^1 \frac{P(X_i|Y_i)P(Y_i)P(X_2|X_i)}{P(X_i)} \right] \right)} \\ \frac{P(X_3|Y_3) \prod_{i=1,2,4,5} \left[\sum_{X_i=0}^1 P(X_i|Y_i)P(Y_i)P(X_3|X_i) \right]}{P(X_3)^2 \sum_{X_3=0}^1 \left(\frac{P(X_3|Y_3)}{P(X_3)^2} \prod_{i=1,2,4,5} \left[\sum_{X_i=0}^1 \frac{P(X_i|Y_i)P(Y_i)P(X_3|X_i)}{P(X_i)} \right] \right)} \\ \frac{P(X_4|Y_4) \prod_{i=1,2,3,5} \left[\sum_{X_i=0}^1 P(X_i|Y_i)P(Y_i)P(X_4|X_i) \right]}{P(X_4)^2 \sum_{X_4=0}^1 \left(\frac{P(X_4|Y_4)}{P(X_4)^2} \prod_{i=1,2,3,5} \left[\sum_{X_i=0}^1 \frac{P(X_i|Y_i)P(Y_i)P(X_4|X_i)}{P(X_i)} \right] \right)} \\ \frac{P(X_5|Y_5) \prod_{i=1,2,3,4} \left[\sum_{X_i=0}^1 P(X_i|Y_i)P(Y_i)P(X_5|X_i) \right]}{P(X_5)^2 \sum_{X_5=0}^1 \left(\frac{P(X_5|Y_5)}{P(X_5)^2} \prod_{i=1,2,3,4} \left[\sum_{X_i=0}^1 \frac{P(X_i|Y_i)P(Y_i)P(X_5|X_i)}{P(X_i)} \right] \right)} \end{array} \right] \quad (8)$$

This method can be used to determine the update function of any network involving loops.

5.2.2 Examining Fixed Points of T and Contractions

The update function T can be used in order to analyze and explain the structure of network graphs that prove to converge to a correct solution under the belief propagation algorithm. Also, the behaviour of convergence can be analyzed as certain properties of networks are varied.

It is important to investigate the specific structures of networks in which the update function, $T(x)$ is a contraction within a small neighbourhood around a fixed point B^* . A contraction $T : X \rightarrow X$, where X is a Banach space, has a smaller image space than pre-image space (range) around a point B^* . The effect of the update function is illustrated in Figure 8. T will always have a fixed point, ie. $B^* = T(B^*)$. From our preliminary analysis, we have found that at least one fixed point exists when all nodes are deterministic and agree. It is important to note that the existence of a fixed point does not imply convergence. Certain fixed points may be unstable. T will not contract locally around unstable fixed points [12].

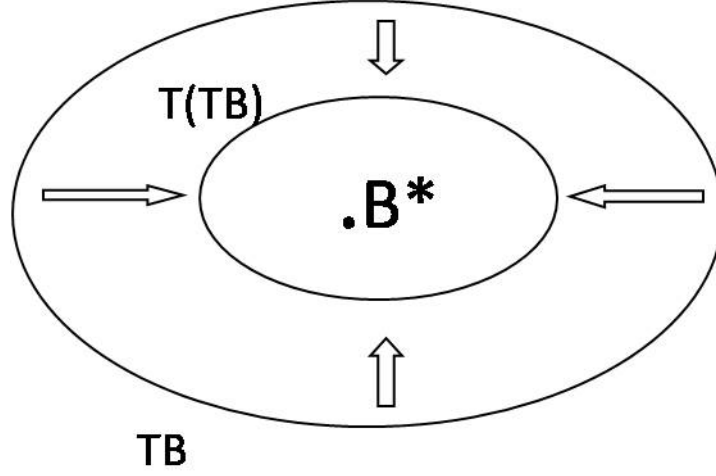


Figure 8: Local contraction around a fixed point B^* . Specifically, this is the effect of the update function $T(x)$ in a small neighbourhood around B^* .

5.2.3 Fixed Point Argument

It is important to note that the transformation T will always have a fixed point. The example described below demonstrates this fixed point concept for the three-node interconnected network:

Let $P(X_i = 0) = 0$ and $P(X_i = 1) = b_i = 1, \forall i \in 1, 2, 3$. We can show a fixed point for the belief at node 1, b_1

$$b_1 = T(b_1, b_2, b_3)$$

$$\begin{aligned}
&= \frac{\sum_{X_2=0}^1 b_2 P(Y_2) P(X_1|X_2) \sum_{X_3=0}^1 b_3 P(Y_3) P(X_1|X_3)}{[P(X_1)]^2 \sum_{X_1=0}^1 \frac{b_1}{[P(X_1)]^2} \left[\sum_{X_2=0}^1 b_2 P(Y_2) P(X_1|X_2) \right] \left[\sum_{X_3=0}^1 b_3 P(Y_3) P(X_1|X_3) \right]} \\
&= \frac{[0+P(Y_2)P(X_1|X_2=1)][0+P(Y_3)P(X_1|X_3=1)]}{[P(X_1=1)]^2 \frac{1}{[P(X_1=1)]} [0+P(Y_2)P(X_1|X_2=1)][0+P(Y_3)P(X_1|X_3=1)]} \\
&= \frac{[P(Y_2)P(X_1|X_2=1)][P(Y_3)P(X_1|X_3=1)]}{1^2 \left[\frac{1}{1^2} \right] [P(Y_2)P(X_1|X_2=1)][P(Y_3)P(X_1|X_3=1)]} \\
&= 1
\end{aligned}$$

(9)

Similarly, we can show that $b_i = P(X_i = 1) = 1$, and $P(X_i = 0) = 1 - b_i = 0$, $\forall i \leq n \in [2, 3]$

This implies that $B_t = TB_{t-1}$, indicating that $[1,1,1]$ is a fixed point. We will later show the existence of other fixed points.

5.2.4 Contraction Argument

Banach's Fixed Point Theorem states that, in general, within a complete metric space, if $T : X \rightarrow X$ is a contraction and if there exists a $B \in X$ such that $T(B^*) = B^*$, then $T^n(B)$ will converge to B^* as $n \rightarrow \infty$. That is for all $B \in [0, 1]^n$, the sequence of iterates of T given by $(B, TB, T(T(B)), T(T(T(B))), \dots)$ converges to the point B^* within a Banach space [12].

More precisely,

If $B \in \{x : \|x - B^*\| \leq \delta\}$, where B^* is a fixed point and if $\|T(B) - T(B^*)\| \leq \rho \|B - B^*\|$, for some $0 \leq \rho < 1$ Then T is a contraction and $[T^n(B)]$ converge to B^* .

Proof:

Let T be a contraction, where $B^* = T(B^*)$ and $\rho \leq 1$ so

$$\begin{aligned} \|T(B) - T(B^*)\| &\leq \rho \|B - B^*\| \\ \|T(B) - B^*\| &\leq \rho \|B - B^*\| \\ \|T^2(B) - B^*\| &\leq \rho^2 \|B - B^*\| \\ \|T^n(B) - B^*\| &\leq \rho^n \|B - B^*\| \\ \text{Hence } [T^n(B)] &\text{ converges to } B^* \end{aligned}$$

(10)

5.2.5 Linearizing

To analyze the contraction properties of T we will use linear approximation. If the linearized T mapping is a contraction, then this implies that T is a contraction within a small region around B^* . This method of linearization is valid because T is both continuous and differentiable.

5.2.6 Norm Discussion

To show convergence of the belief propagation algorithm, we must show that the T mapping is a contraction in our discrete time space. A number of norms can be used to investigate T , such as the L^1 norm and the L^∞ norm, which are described below for a vector X with entries $x_1, x_2, x_3, \dots, x_n$.

L^1 norm:

$$\|X\|_1 = \sum_i |x_i|, i = 1, \dots, n$$

L^∞ norm:

$$\|X\|_\infty = \max(x_i), i = 1, \dots, n$$

Our investigation of the contraction properties of T will require a complete normed space. We will consider R^n and two norms, L^∞ norm and L^1 which are seen above. Under both of these norms R^n is a complete space. This means that Banach's fixed point theorem holds. For the purposes of our analysis, we will only look at these two norms. It is easy to see that our analysis and results hold under the general L^p norm for any positive integer p . Our analysis can be extended to include all L^p norms because all possible belief values are positive and less than one. So, our restriction to the two stated norms is without loss of generality. The general L^p norm is seen below.

L^p norm

$$\|X\|_p = (\sum_i |x_i|^p)^{1/p}, i = 1, \dots, n$$

6 RESULTS

(ALL MATLAB CODE IS CONTAINED IN APPENDIX C)

In this section we will present our proof of convergence and existence of fixed points. Results are from mathematical analysis using pencil and paper, Maple and three MATLAB simulations, one for each analyzed network (a Markov network, a tree network and a fully connected network).

6.1 Convergence of Markov and Tree-Structured Networks in MATLAB Simulation

In both the Markov model and the tree structured networks, a computer simulation was created using MATLAB in order to test the algorithm and generated equations as well as to confirm analytical results. These simulations provided a more thorough understanding of the algorithm's process of updating beliefs under different structures that were known to converge. This information assisted in formulating the approach for analysis which was used in the design portion of our project. The computer simulations confirmed that the algorithm converged to the correct answer in both the Markov structured network and the tree structured network.

An example of the output values from the Markov model simulation is displayed in section Equation 9. The output shows that this Markov chain converges after nine iterations. For this simulation, our network was set with the following parameters: $P(X_i = 1) = 0.5$, $P(X_i = k|Y_i = k) = 0.75$ and initial beliefs $P(X_i = k|Y_i = k) = 0.75$. Also, Y -values were randomly generated with the use of MATLAB.

Iteration	P(X₁=1)	P(X₂=1)	P(X₃=1)
1	0.7570	0.4989	0.5806
2	0.8169	0.5602	0.7489
3	0.8371	0.5808	0.8140
4	0.8437	0.5874	0.8362
5	0.8458	0.5895	0.8434
6	0.8464	0.5902	0.8457
7	0.8467	0.5904	0.8464
8	0.8467	0.5905	0.8467
9	0.8468	0.5905	0.8467
10	0.8468	0.5905	0.8467

Corresponding Y values	1	0	1
-------------------------------	----------	----------	----------

Figure 9: Tree node Markov Chain MATLAB simulation output

6.2 Convergence in a Fully Connected Network

In this subsection we will discuss the process of finding fixed points and the properties of convergence around such fixed points for a fully connected network. It should be noted that although other structures were not analyzed in this report this analysis can be extended to include various network structures.

6.2.1 Determining Fixed Points

The MATLAB simulation for the fully connected network was used in order to determine if any fixed points exist under the update function at specific parameterizations. The simulation was successful in finding fixed points but the algorithm tended to converge on the outer regions (when the belief of each node is zero or one). To further our analysis, it became necessary to find other, non-trivial fixed points. To solve this problem the beliefs were manually initialized. The system of equations resulting from $B = TB$, where B is the belief vector and T is the transformation, was solved under different parameterizations using Maple. A summary of fixed point results can also be seen in Appendix B.3.

6.2.2 Showing Convergence To Fixed Points

The next step in our procedure is to show that T converges in a small neighbourhood around the determined fixed point B^* . Fixed points were found by solving $B = TB$ with the aid of Maple software, and T was then tested for local convergence by setting initial beliefs close to the fixed point. It was found that there could be many points to which the algorithm successfully converges under a given parameterization of input variables. In addition, many fixed points were found that could only be reached if they were set as the initial network belief. This implied that the algorithm does not necessarily converge locally around fixed points, which led to a closer look at the contraction properties of T around various fixed points.

Using the linear approximation of T around the fixed points, we were able to further examine the update function's contraction properties around specific fixed points. An example of a fixed point in the three-node fully connected loop, as found from the procedure explained in section 6.2.1 can be seen below. In this case, T is a contraction within a close region of the fixed point.

Let there be three nodes, in a fully connected network, and let B^* be the fixed point.

$$B^* = \begin{bmatrix} 0.2656820273 \\ 0.2656820273 \\ 0.2656820273 \end{bmatrix} \tag{11}$$

With the help of Maple we found the Jacobian of T evaluated at B^* . For this particular example, the Jacobian evaluated at B^* is given by:

$$\begin{bmatrix} 0.50695920650 & 0.03203727592 & 0.03203727592 \\ 0.03203727592 & 0.50695920650 & 0.03203727592 \\ 0.03203727592 & 0.03203727592 & 0.50695920650 \end{bmatrix} \quad (12)$$

A proof of this contraction property for the example stated above can be seen below:

Proof of contraction property $\|T(B_o) - T(B^*)\| \leq \rho \|B_o - B^*\|$ under L^1 norm, for fixed B^* and initial belief vector $B_0 = B^* + [\epsilon_1, \epsilon_2, \epsilon_3]^T$

$$\begin{aligned} \|T(B_0) - T(B^*)\|_1 &= \left\| \begin{bmatrix} 0.507\epsilon_1 + 0.032\epsilon_2 + 0.032\epsilon_3 \\ 0.032\epsilon_1 + 0.507\epsilon_2 + 0.032\epsilon_3 \\ 0.032\epsilon_1 + 0.032\epsilon_2 + 0.507\epsilon_3 \end{bmatrix} \right\|_1 \\ &= |0.507\epsilon_1 + 0.032\epsilon_2 + 0.032\epsilon_3| + |0.032\epsilon_1 + 0.507\epsilon_2 + 0.032\epsilon_3| + |0.032\epsilon_1 + 0.032\epsilon_2 + 0.507\epsilon_3| \\ &\leq |0.507\epsilon_1| + |0.032\epsilon_2| + |0.032\epsilon_3| + |0.032\epsilon_1| + |0.507\epsilon_2| + |0.032\epsilon_3| + \\ &\quad |0.032\epsilon_1| + |0.032\epsilon_2| + |0.507\epsilon_3| \\ &= 0.507|\epsilon_1| + 0.032|\epsilon_2| + 0.032|\epsilon_3| + 0.032|\epsilon_1| + 0.507|\epsilon_2| + 0.032|\epsilon_3| + \\ &\quad 0.032|\epsilon_1| + 0.032|\epsilon_2| + 0.507|\epsilon_3| \\ &\leq 0.58|\epsilon_1| + 0.58|\epsilon_2| + 0.58|\epsilon_3| \\ &= 0.58(|\epsilon_1| + |\epsilon_2| + |\epsilon_3|) \\ &= 0.58 \left\| \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} \right\|_1 \\ &\leq \rho \|B_0 - B^*\|_1 \text{ for some } 0 \leq \rho < 1 \end{aligned} \quad (13)$$

This shows that T is a contraction within a small region of the fixed point. This is also true for the L^∞ norm as seen below.

$$\begin{aligned}
\|T(B_0) - T(B^*)\|_\infty &= \left\| \begin{bmatrix} 0.507\epsilon_1 + 0.032\epsilon_2 + 0.032\epsilon_3 \\ 0.032\epsilon_1 + 0.507\epsilon_2 + 0.032\epsilon_3 \\ 0.032\epsilon_1 + 0.032\epsilon_2 + 0.507\epsilon_3 \end{bmatrix} \right\|_\infty \\
&= \max(|0.507\epsilon_1 + 0.032\epsilon_2 + 0.032\epsilon_3|, \\
&\quad |0.032\epsilon_1 + 0.507\epsilon_2 + 0.032\epsilon_3|, |0.032\epsilon_1 + 0.032\epsilon_2 + 0.507\epsilon_3|) \\
&\leq \max(|0.507\epsilon_1| + |0.032\epsilon_2| + |0.032\epsilon_3|, (|0.032\epsilon_1| + |0.507\epsilon_2| + |0.032\epsilon_3|), \\
&\quad (|0.032\epsilon_1| + |0.032\epsilon_2| + |0.507\epsilon_3|)) \\
&= \max[(0.507|\epsilon_1| + 0.032|\epsilon_2| + 0.032|\epsilon_3|), (0.032|\epsilon_1| + 0.507|\epsilon_2| + 0.032|\epsilon_3|), \\
&\quad (0.032|\epsilon_1| + 0.032|\epsilon_2| + 0.507|\epsilon_3|)]
\end{aligned}$$

WLOG assume that $\epsilon_i = \max(\epsilon_1, \epsilon_2, \epsilon_3)$, implying that :

$$\begin{aligned}
&\leq 0.507|\epsilon_i| + 0.032|\epsilon_i| + 0.032|\epsilon_i| \\
&= 0.58|\epsilon_i| \\
&= 0.58 \left\| \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} \right\|_\infty \\
&\leq \rho \|B_0 - B^*\|_\infty \text{ for some } 0 \leq \rho < 1
\end{aligned}$$

(14)

6.3 Extending Our Analysis

The procedure that we've used in order to draw conclusions for the three-node fully connected loop can be extended to networks of various structures. In addition to our rigorous analysis on the three-node loop, we've also analyzed the fully-connected five-node loop from Figure 7 under the same procedure. The Matlab code used to find fixed points as well as to linearize the system around such fixed points can be seen in Appendix C.

7 DISCUSSION

Over the course of our project, we created a new systematic approach for analyzing convergence of the loopy belief propagation algorithm. The belief propagation algorithm was analyzed under a three-node loopy network, and the generated method of analyzing the algorithm under loopy systems for local convergence proved to be successful.

The developed method consists of determining a transfer function, which is then used to find a fixed point. The system is then linearized around a fixed point and this result is used to show existence of a local contraction. The successful proof the existence of a continuous contraction can be used to determine the amount of error allowable in initial beliefs that will allow the algorithm to converge. This implies that given a set range of allowable error in initial beliefs, the algorithm will converge for loopy systems.

Our research has shown that fixed beliefs are present in all loopy three-node networks, and they depend on predefined parameters. However, in certain cases, there may not exist a fixed point under which the algorithm converges locally. A detailed table of fixed points found for loopy graphs with varying parameters (such as distributions for variables and dependence on neighbouring beliefs) is included in Appendix B.3.

The created methodology for mathematical analysis proved to display promising results. Preliminary research indicated that the algorithm will always converge under Markov chains or tree-structured networks. Generated MATLAB simulations confirm convergence in these cases after a finite number of iterations.

These simulations utilized random number generators in order to set and vary the observations. It was then determined that the update of beliefs at time t could be modelled as a function of previous beliefs at time $t - 1$. This function, which was defined as a matrix T , was then used in order to find the existence of fixed points in the system via the use of a Maple program.

In order to demonstrate local convergence, the system was linearized around a chosen fixed point. The result was then used to prove the existence of a

local contraction around the fixed point under different norms. This indicates that the algorithm generates positive results for the loopy case given a certain level of disagreement, or error, in original beliefs.

8 CONCLUSION

8.1 Concluding Remarks

The original methodology created and analyzed in this project is used to determine if each node will reach a conclusion -or if the belief propagation algorithm will converge- after a finite number of iterations. We have displayed the existence of a small neighbourhood around a fixed point where the algorithm will converge in loopy networks. This result, as well as the methodology that was used, has been generated independently and, to our knowledge, has not been seen in this form in previous research papers.

To be more specific, our methodology has been proven to show the existence of a continuous local contraction around a fixed point in loopy networks.

In order to obtain this result, we first needed to find a fixed point. To do this, concepts of probability were used to determine the general form of a transfer function when a system involves loops. This transfer function evaluated at an arbitrary point was set to be equal to the same arbitrary point, and the resulting system of equations was solved in order to find a fixed point of the transformation. The system was then linearized around the fixed point and the result was used in order to show the existence of a local contraction.

We have proven that the algorithm will converge in a loopy network given a small amount of error in initial probabilities. More generally, given a initial nodal beliefs within a sufficiently small range of the fixed point, the belief propagation algorithm will converge to a fixed point in loopy systems. We have also managed to successfully extend the set of models to which the algorithm can be used to infer accurate results.

In addition to our generated method of mathematical analysis, we've created MATLAB simulations which experimentally confirm results that were

proven mathematically. The simulations were created for four networks: a Markov chain, a tree-structure, a three node loop, and a fully-connected five node network. Random number generators are used to create initial beliefs in the simulations for the Markov chain and the tree structure. In each case, as these simulations are run multiple times with varying initial beliefs, the program illustrates convergence.

In simulations for the three node loop and fully connected five node network, initial beliefs are set to be within a small neighbourhood of generated fixed points. Again, the simulations confirm the results seen from our mathematical methodology as convergence occurs in these cases as well.

8.2 Future Work

For networks providing update functions with fixed points, it is not always the case that there exists a contraction around such fixed points. However, the demonstration of the update function, T , being a contraction within a local neighbourhood of an example fixed point, as shown in Section 6 shows that convergence of the belief propagation algorithm can indeed occur in graphs with loopy structures. Further research in this area could be performed in order to examine the initial parameterization of variables that would result in fixed points to which beliefs in local neighbourhoods will always converge. Investigation into the size and boundaries of these local neighbourhoods could also be performed.

References

- [1] Michael Ghil Andrew W. Robertson Padhraic Smyth Alexander T. Ihler, Sergey Kirshner. Graphical models for statistical inference and data assimilation. *Physica D*, 2006.
- [2] Kevin P. Murphy. An introduction to graphical models. May 2001.
- [3] Balaji Prabhakar Devavrat Shah Stephen Boyd, Arpita Ghosh. Gossip algorithms: Design, analysis, and applications. 2005.
- [4] William T. Freeman Yair Weiss. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *Information Theory, IEEE Transactions on*, 47(2):736–744, Feb 2001.
- [5] Shigeru Mase Nobuyuki Taga. On the convergence of belief propagation algorithm for stochastic networks with loops. March 2004.
- [6] William T. Freeman Jonathan S. Yedidia and Yair Weiss. Understanding belief propagation and its generalizations. 2002.
- [7] Brendan J. Frey Hans-Andrea Loeliger Frank R. Kschichang. Factor graphs and the sum-product algorithm. *Transaction on Information Theory*, 47(2), Feb 2001.
- [8] Alex Olshevky John N. Tsitsiklis Vincent D. Blondel, Julien M. Hendrickx. Convergence in multiagent coordination, consensus, and flocking. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, Dec 2005.
- [9] Buyurman Baykal Muhammet Fatih Bayramoglu, Ali Ozgur Yilmaz. Sub graph approach in iterative sum-product algorithm. 2006.
- [10] Michael I. Jordan Kevin P. Murphy, Yair Weiss. Loopy belief propagation for approximate inference: An empirical study. 1999.
- [11] Casey Boardman. Pearl’s belief propagation algorithm and loopy bayesian networks. Apr 2004.
- [12] Joris M. Mooij and Hilbert J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. May 2007.

A APPENDIX - Proofs, Derivations and Long Equations

A.1 Markov Derivation

Derivation of the Markov Update equation for the network illustrated in Figure 4.

$$\begin{aligned}
P(X_3|Y_1Y_2Y_3) &= \sum_{X_2=0}^1 \sum_{X_1=0}^1 P(X_1X_2X_3|Y_1Y_2Y_3) \\
&= \frac{\sum_{X_2=0}^1 \sum_{X_1=0}^1 P(X_1X_2X_3Y_1Y_2|Y_3)}{P(Y_1Y_2|Y_3)} \\
&= \frac{\sum_{X_2=0}^1 \sum_{X_1=0}^1 P(Y_1|X_1X_2X_3Y_2Y_3)P(X_1|X_2X_3Y_2Y_3)P(Y_2|X_2X_3Y_3)P(X_2|X_3Y_3)P(X_3|Y_3)}{\sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 P(X_1X_2X_3Y_1Y_2|Y_3)} \\
&= \frac{\sum_{X_2=0}^1 \sum_{X_1=0}^1 \left(\frac{P(X_1|Y_1)P(Y_1)}{P(X_1)} \right) \left(\frac{P(X_2|X_1)P(X_1)}{P(X_2)} \right) \left(\frac{P(X_2|Y_2)P(Y_2)}{P(X_2)} \right) \left(\frac{P(X_3|X_2)P(X_2)}{P(X_3)} \right) P(X_3|Y_3)}{\sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 P(Y_1|X_1X_2X_3Y_2Y_3)P(X_1|X_2X_3Y_2Y_3)P(Y_2|X_2X_3Y_3)P(X_2|X_3Y_3)P(X_3|Y_3)} \\
&= \frac{\frac{P(X_3|Y_3)}{P(X_3)} \sum_{X_2=0}^1 \left(P(X_2|Y_2)P(Y_2)P(X_3|X_2) \sum_{X_1=0}^1 P(X_1|Y_1)P(Y_1)P(X_2|X_1) \right)}{\sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 \left(\frac{P(X_1|Y_1)P(Y_1)}{P(X_1)} \right) \left(\frac{P(X_2|X_1)P(X_1)}{P(X_2)} \right) \left(\frac{P(X_2|Y_2)P(Y_2)}{P(X_2)} \right) \left(\frac{P(X_3|X_2)P(X_2)}{P(X_3)} \right) P(X_3|Y_3)} \\
&= \frac{P(X_3|Y_3) \sum_{X_2=0}^1 \left(P(X_2|Y_2)P(Y_2)P(X_3|X_2) \sum_{X_1=0}^1 P(X_1|Y_1)P(Y_1)P(X_2|X_1) \right)}{\sum_{X_3=0}^1 \left(\frac{P(X_3|Y_3)}{P(X_3)} \sum_{X_2=0}^1 \left(P(X_3|X_2) \frac{P(X_2|Y_2)P(Y_2)}{P(X_2)} \sum_{X_1=0}^1 P(X_2|X_1)P(X_1|Y_1)P(Y_1) \right) \right)}
\end{aligned} \tag{15}$$

A.2 Tree-Structure Derivation

The derivation for the upward sweep of the tree structured diagram in Figure 4.1

$$\begin{aligned}
P(X_1|Y_1Y_2Y_3) &= \sum_{X_2=0}^1 \left(\sum_{X_3=0}^1 P(X_1X_2X_3|Y_1Y_2Y_3) \right) \\
&= \frac{\sum_{X_2=0}^1 \left(\sum_{X_3=0}^1 P(X_1X_2X_3Y_2Y_3|Y_1) \right)}{P(Y_2Y_3|Y_1)}
\end{aligned}$$

$$\begin{aligned}
&= \frac{\sum_{X_2=0}^1 \left(\sum_{X_3=0}^1 P(Y_3|X_3 X_2 X_1 Y_2 Y_1) P(X_3|X_2 X_1 Y_2 Y_1) P(Y_2|X_2 X_1 Y_1) P(X_3|X_1 Y_1) P(X_1|Y_1) \right)}{\sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 P(X_1 X_2 X_3 Y_2 Y_3 | Y_1)} \\
&= \frac{\sum_{X_2=0}^1 \left(\sum_{X_3=0}^1 P(Y_2|X_2) P(X_2|X_1) P(Y_3|X_3) P(X_3|X_1) P(X_1|Y_1) \right)}{\sum_{X_1=0}^1 \sum_{X_2=0}^1 \left(\sum_{X_3=0}^1 P(X_1 X_2 X_3 Y_2 Y_3 | Y_1) \right)} \\
&= \frac{\sum_{X_2=0}^1 \sum_{X_3=0}^1 \frac{P(X_3|Y_3) P(Y_3)}{P(X_3)} \frac{P(X_1|X_3) P(X_3)}{P(X_1)} \frac{P(X_2|Y_2) P(Y_2)}{P(X_2)} \frac{P(X_1|X_2) P(X_2)}{P(X_1)} P(X_1|Y_1)}{\sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 (P(Y_3|X_3 X_2 X_1 Y_2 Y_1) P(X_3|X_2 X_1 Y_2 Y_1) P(Y_2|X_2 X_1 Y_1) P(X_2|X_1 Y_1) P(X_1|Y_1))} \\
&= \frac{\frac{P(X_1|Y_1)}{P(X_1)^2} \sum_{X_3=0}^1 P(X_3|Y_3) P(Y_3) P(X_1|X_3) \sum_{X_2=0}^1 P(X_2|Y_2) P(Y_2) P(X_1|X_2)}{\sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 \left(\frac{P(X_3|Y_3) P(Y_3)}{P(X_3)} \frac{P(X_1|X_3) P(X_3)}{P(X_1)} \frac{P(X_2|Y_2) P(Y_2)}{P(X_2)} \frac{P(X_1|X_2) P(X_2)}{P(X_1)} P(X_1|Y_1) \right)} \\
&= \frac{\frac{P(X_1|Y_1)}{P(X_1)^2} \sum_{X_3=0}^1 P(X_3|Y_3) P(Y_3) P(X_1|X_3) \sum_{X_2=0}^1 P(X_2|Y_2) P(Y_2) P(X_1|X_2)}{\sum_{X_1=0}^1 \left(\frac{P(X_1|Y_1)}{P(X_1)^2} \sum_{X_3=0}^1 P(X_3|Y_3) P(Y_3) P(X_1|X_3) \sum_{X_2=0}^1 P(X_2|Y_2) P(Y_2) P(X_1|X_2) \right)}
\end{aligned}$$

(16)

B APPENDIX - Maple code and Results

In order to determine fixed points in the system, a Maple program was created. The update function at time t was set equal to the beliefs at time $t - 1$. Maple was used to solve the resulting system of equations for various networks. In addition, T was then linearized around the determined fixed points in order to produce the Jacobian of T evaluated at the fixed point. This result was then used for further analysis.

B.1 Maple Code for the Three-Node Fully-Connected Loop

This code is utilized to find fixed points in the 3 node loopy system as well as to evaluate the Jacobian of T at a chosen fixed point.

>

`with(RealDomain); with(linalg);`

>

$$P(Y_{1=1}) := \frac{3}{8}; P(Y_{2=1}) := \frac{3}{8}; P(Y_{3=1}) := \frac{3}{8}; P(X_{1=1}) := \frac{3}{8}; P(X_{2=1}) := \frac{3}{8};$$

$$P(X_{3=1}) := \frac{3}{8}; P(X_{1=0}GX_{2=1}) := 0.6; P(X_{1=0}GX_{2=0}) := 1 - P(X_{1=0}GX_{2=1});$$

$$P(Y_{1=1}) := \frac{3}{8}$$

$$P(Y_{2=1}) := \frac{3}{8}$$

$$P(Y_{3=1}) := \frac{3}{8}$$

$$P(X_{1=1}) := \frac{3}{8}$$

$$P(X_{2=1}) := \frac{3}{8}$$

$$P(X_{3=1}) := \frac{3}{8}$$

$$P(X_{1=0}GX_{2=1}) := 0.6$$

$$P(X_{1=0}GX_{2=0}) := 0.4$$

>

$$\begin{aligned} P(X_{1=1}GX_{2=0}) &:= P(X_{1=0}GX_{2=1}); P(X_{3=1}GX_{2=0}) := P(X_{1=0}GX_{2=1}); \\ P(X_{3=0}GX_{1=1}) &:= P(X_{1=0}GX_{2=1}); P(X_{3=1}GX_{1=0}) := P(X_{1=0}GX_{2=1}); \\ P(X_{3=0}GX_{2=1}) &:= P(X_{1=0}GX_{2=1}); P(X_{2=1}GX_{3=0}) := P(X_{1=0}GX_{2=1}); \\ P(X_{2=0}GX_{3=1}) &:= P(X_{1=0}GX_{2=1}); P(X_{1=0}GX_{3=1}) := P(X_{1=0}GX_{2=1}); \\ P(X_{1=1}GX_{3=0}) &:= P(X_{1=0}GX_{2=1}); P(X_{2=0}GX_{1=1}) := P(X_{1=0}GX_{2=1}); \\ P(X_{2=1}GX_{1=0}) &:= P(X_{1=0}GX_{2=1}); P(X_{1=1}GX_{3=1}) := P(X_{1=0}GX_{2=0}); \\ P(X_{2=0}GX_{1=0}) &:= P(X_{1=0}GX_{2=0}); P(X_{3=1}GX_{1=1}) := P(X_{1=0}GX_{2=0}); \\ P(X_{3=0}GX_{1=0}) &:= P(X_{1=0}GX_{2=0}); P(X_{2=0}GX_{3=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{1=0}GX_{3=0}) &:= P(X_{1=0}GX_{2=0}); P(X_{2=1}GX_{1=1}) := P(X_{1=0}GX_{2=0}); \\ P(X_{2=1}GX_{3=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{3=0}GX_{2=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{3=1}GX_{2=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{1=1}GX_{2=1}) := P(X_{1=0}GX_{2=0}); \end{aligned}$$

$$\begin{aligned} P(X_{1=1}GX_{2=0}) &:= 0.6 \\ P(X_{3=1}GX_{2=0}) &:= 0.6 \\ P(X_{3=0}GX_{1=1}) &:= 0.6 \\ P(X_{3=1}GX_{1=0}) &:= 0.6 \\ P(X_{3=0}GX_{2=1}) &:= 0.6 \\ P(X_{2=1}GX_{3=0}) &:= 0.6 \\ P(X_{2=0}GX_{3=1}) &:= 0.6 \\ P(X_{1=0}GX_{3=1}) &:= 0.6 \\ P(X_{1=1}GX_{3=0}) &:= 0.6 \\ P(X_{2=0}GX_{1=1}) &:= 0.6 \\ P(X_{2=1}GX_{1=0}) &:= 0.6 \\ P(X_{1=1}GX_{3=1}) &:= 0.4 \\ P(X_{2=0}GX_{1=0}) &:= 0.4 \\ P(X_{3=1}GX_{1=1}) &:= 0.4 \\ P(X_{3=0}GX_{1=0}) &:= 0.4 \\ P(X_{2=0}GX_{3=0}) &:= 0.4 \\ P(X_{1=0}GX_{3=0}) &:= 0.4 \\ P(X_{2=1}GX_{1=1}) &:= 0.4 \\ P(X_{2=1}GX_{3=1}) &:= 0.4 \\ P(X_{3=0}GX_{2=0}) &:= 0.4 \\ P(X_{3=1}GX_{2=1}) &:= 0.4 \\ P(X_{1=1}GX_{2=1}) &:= 0.4 \end{aligned}$$

>

$$P(Y_{1=0}) := 1 - P(Y_{1=1}); P(Y_{2=0}) := 1 - P(Y_{2=1}); P(Y_{3=0}) := 1 - P(Y_{3=1});$$

$$P(X_{1=0}) := 1 - P(X_{1=1}); P(X_{2=0}) := 1 - P(X_{2=1}); P(X_{3=0}) := 1 - P(X_{3=1});$$

$$P(Y_{1=0}) := \frac{5}{8}$$

$$P(Y_{2=0}) := \frac{5}{8}$$

$$P(Y_{3=0}) := \frac{5}{8}$$

$$P(X_{1=0}) := \frac{5}{8}$$

$$P(X_{2=0}) := \frac{5}{8}$$

$$P(X_{3=0}) := \frac{5}{8}$$

>

$$b_{1=0} := 1 - b_{1=1}; b_{2=0} := 1 - b_{2=1}; b_{3=0} := 1 - b_{3=1};$$

$$b_{1=0} := 1 - b_{1=1}$$

$$b_{2=0} := 1 - b_{2=1}$$

$$b_{3=0} := 1 - b_{3=1}$$

>

$$t1 := \text{simplify} \left(\text{expand} \left(\left(b_{1=1} \cdot \left(\sum_{i=0}^1 (P(Y_{2=1}) \cdot b_{2=i} \cdot P(X_{1=1}GX_{2=i})) \right) \right) \right. \right.$$

$$\cdot \left. \left(\sum_{j=0}^1 (P(Y_{3=1}) \cdot b_{3=j} \cdot P(X_{1=1}GX_{3=j})) \right) \right) \left/ \left((P(X_{1=1}))^2 \right. \right.$$

$$\cdot \left. \left(\sum_{k=0}^1 \left(\frac{b_{1=k}}{(P(X_{1=k}))^2} \cdot \left(\sum_{l=0}^1 (P(X_{1=k}GX_{2=l}) \cdot b_{2=l} \cdot P(Y_{2=l})) \right) \right) \right) \right.$$

$$\cdot \left. \left(\sum_{m=0}^1 (P(X_{1=k}GX_{3=m}) \cdot b_{3=m} \cdot P(Y_{3=m})) \right) \right) \right);$$

$$t1 := (25 \cdot b_{1=1} (9 - 3 \cdot b_{3=1} - 3 \cdot b_{2=1} + b_{2=1} b_{3=1})) / (100 - 10 \cdot b_{3=1} - 10 \cdot b_{2=1}$$

$$+ b_{2=1} b_{3=1} + 525 \cdot b_{1=1} - 365 \cdot b_{1=1} b_{3=1} - 365 \cdot b_{1=1} b_{2=1}$$

$$+ 224 \cdot b_{1=1} b_{2=1} b_{3=1})$$

>

$$\begin{aligned} t2 := & \text{simplify} \left(\text{expand} \left(\left(b_{2=1} \cdot \left(\sum_{i=0}^1 (P(Y_{1=1}) \cdot b_{1=i} \cdot P(X_{2=1}GX_{1=i})) \right) \right) \right. \right. \\ & \cdot \left. \left(\sum_{j=0}^1 (P(Y_{3=1}) \cdot b_{3=j} \cdot P(X_{2=1}GX_{3=j})) \right) \right) \Big/ \left((P(X_{2=1}))^2 \right. \\ & \cdot \left. \left(\sum_{k=0}^1 \left(\frac{b_{2=k}}{(P(X_{2=k}))^2} \cdot \left(\sum_{l=0}^1 (P(X_{2=k}GX_{1=l}) \cdot b_{1=l} \cdot P(Y_{1=l})) \right) \right) \right) \right. \\ & \left. \left. \left. \left. \left. \left. \left(\sum_{m=0}^1 (P(X_{2=k}GX_{3=m}) \cdot b_{3=m} \cdot P(Y_{3=m})) \right) \right) \right) \right) \right) \right); \end{aligned}$$

$$\begin{aligned} t2 := & (25 \cdot b_{2=1} (9 \cdot -3 \cdot b_{3=1} - 3 \cdot b_{1=1} + b_{1=1} b_{3=1})) / (100 \cdot -10 \cdot b_{3=1} - 10 \cdot b_{1=1} \\ & + b_{1=1} b_{3=1} + 525 \cdot b_{2=1} - 365 \cdot b_{2=1} b_{3=1} - 365 \cdot b_{1=1} b_{2=1} \\ & + 224 \cdot b_{1=1} b_{2=1} b_{3=1}) \end{aligned}$$

>

$$\begin{aligned} t3 := & \text{simplify} \left(\text{expand} \left(\left(b_{3=1} \cdot \left(\sum_{i=0}^1 (P(Y_{2=1}) \cdot b_{2=i} \cdot P(X_{3=1}GX_{2=i})) \right) \right) \right. \right. \\ & \cdot \left. \left(\sum_{j=0}^1 (P(Y_{1=1}) \cdot b_{1=j} \cdot P(X_{3=1}GX_{1=j})) \right) \right) \Big/ \left((P(X_{3=1}))^2 \right. \\ & \cdot \left. \left(\sum_{k=0}^1 \left(\frac{b_{3=k}}{(P(X_{3=k}))^2} \cdot \left(\sum_{l=0}^1 (P(X_{3=k}GX_{2=l}) \cdot b_{2=l} \cdot P(Y_{2=l})) \right) \right) \right) \right. \\ & \left. \left. \left. \left. \left. \left. \left(\sum_{m=0}^1 (P(X_{3=k}GX_{1=m}) \cdot b_{1=m} \cdot P(Y_{1=m})) \right) \right) \right) \right) \right) \right); \end{aligned}$$

$$\begin{aligned} t3 := & (25 \cdot b_{3=1} (9 \cdot -3 \cdot b_{1=1} - 3 \cdot b_{2=1} + b_{1=1} b_{2=1})) / (100 \cdot -10 \cdot b_{1=1} - 10 \cdot b_{2=1} \\ & + b_{1=1} b_{2=1} + 525 \cdot b_{3=1} - 365 \cdot b_{1=1} b_{3=1} - 365 \cdot b_{2=1} b_{3=1} \\ & + 224 \cdot b_{1=1} b_{2=1} b_{3=1}) \end{aligned}$$

>

$$T := \begin{bmatrix} t1 \\ t2 \\ t3 \end{bmatrix};$$

$$T := \left[\left[\left(25 \cdot b_{1=1} \left(9 \cdot -3 \cdot b_{3=1} - 3 \cdot b_{2=1} + b_{2=1} b_{3=1} \right) \right) / \left(100 \cdot -10 \cdot b_{3=1} - 10 \cdot b_{2=1} + b_{2=1} b_{3=1} + 525 \cdot b_{1=1} - 365 \cdot b_{1=1} b_{3=1} - 365 \cdot b_{1=1} b_{2=1} + 224 \cdot b_{1=1} b_{2=1} b_{3=1} \right) \right], \right. \\
\left. \left[\left(25 \cdot b_{2=1} \left(9 \cdot -3 \cdot b_{3=1} - 3 \cdot b_{1=1} + b_{1=1} b_{3=1} \right) \right) / \left(100 \cdot -10 \cdot b_{3=1} - 10 \cdot b_{1=1} + b_{1=1} b_{3=1} + 525 \cdot b_{2=1} - 365 \cdot b_{2=1} b_{3=1} - 365 \cdot b_{1=1} b_{2=1} + 224 \cdot b_{1=1} b_{2=1} b_{3=1} \right) \right], \right. \\
\left. \left[\left(25 \cdot b_{3=1} \left(9 \cdot -3 \cdot b_{1=1} - 3 \cdot b_{2=1} + b_{1=1} b_{2=1} \right) \right) / \left(100 \cdot -10 \cdot b_{1=1} - 10 \cdot b_{2=1} + b_{1=1} b_{2=1} + 525 \cdot b_{3=1} - 365 \cdot b_{1=1} b_{3=1} - 365 \cdot b_{2=1} b_{3=1} + 224 \cdot b_{1=1} b_{2=1} b_{3=1} \right) \right] \right]$$

>

$$\text{solve}(\{t1 = b_{1=1}, t2 = b_{2=1}, t3 = b_{3=1}\}, [b_{1=1}, b_{2=1}, b_{3=1}]);$$

$$\left[[b_{1=1} = 0., b_{2=1} = 0., b_{3=1} = 0.], [b_{1=1} = 0.2380952381, b_{2=1} = 0., b_{3=1} = 0.], [b_{1=1} = 0., b_{2=1} = 0.2380952381, b_{3=1} = 0.], [b_{1=1} = 0.2507673986, b_{2=1} = 0.2507673986, b_{3=1} = 0.], [b_{1=1} = 1.365670958, b_{2=1} = 1.365670958, b_{3=1} = 0.], [b_{1=1} = 1., b_{2=1} = 1., b_{3=1} = 1.], [b_{1=1} = 0., b_{2=1} = 0., b_{3=1} = 0.2380952381], [b_{1=1} = 0.1283884937, b_{2=1} = 1.348973607, b_{3=1} = 1.348973607], [b_{1=1} = 1.348973607, b_{2=1} = 0.1283884937, b_{3=1} = 1.348973607], [b_{1=1} = 1.348973607, b_{2=1} = 1.348973607, b_{3=1} = 0.1283884937], [b_{1=1} = 0.2507673986, b_{2=1} = 0., b_{3=1} = 0.2507673986], [b_{1=1} = 1.365670958, b_{2=1} = 0., b_{3=1} = 1.365670958], [b_{1=1} = 0., b_{2=1} = 0.2507673986, b_{3=1} = 0.2507673986], [b_{1=1} = 0., b_{2=1} = 1.365670958, b_{3=1} = 1.365670958], [b_{1=1} = 0.2656820273, b_{2=1} = 0.2656820273, b_{3=1} = 0.2656820273], [b_{1=1} = 2.100389401, b_{2=1} = 2.100389401, b_{3=1} = 2.100389401]]$$

>

```

solve( {t1 = b1 = 1, t2 = b2 = 1, t3 = b3 = 1}, [b1 = 1, b2 = 1, b3 = 1]);
[[ [b1 = 1, b2 = 1, b3 = 1], [b1 = 0.2380952381, b2 = 0., b3 = 0.], [b1 = 1.
= 0., b2 = 0.2380952381, b3 = 0.], [b1 = 0.2507673986, b2 = 1
= 0.2507673986, b3 = 0.], [b1 = 1.365670958, b2 = 1.365670958, b3 = 1
= 0.], [b1 = 1., b2 = 1., b3 = 1.], [b1 = 0., b2 = 0., b3 = 1
= 0.2380952381 ], [b1 = 0.1283884937, b2 = 1.348973607, b3 = 1
= 1.348973607 ], [b1 = 1.348973607, b2 = 0.1283884937, b3 = 1
= 1.348973607 ], [b1 = 1.348973607, b2 = 1.348973607, b3 = 1
= 0.1283884937 ], [b1 = 0.2507673986, b2 = 0., b3 = 0.2507673986 ],
[b1 = 1.365670958, b2 = 0., b3 = 1.365670958 ], [b1 = 0., b2 = 1
= 0.2507673986, b3 = 0.2507673986 ], [b1 = 0., b2 = 1.365670958, b3 = 1
= 1.365670958 ], [b1 = 0.2656820273, b2 = 0.2656820273, b3 = 1
= 0.2656820273 ], [b1 = 2.100389401, b2 = 2.100389401, b3 = 1
= 2.100389401 ]]

```

>

$$\text{Jacobian} := \text{simplify} \left(\begin{pmatrix} \frac{\partial}{\partial b_{1=1}} t1 & \frac{\partial}{\partial b_{2=1}} t1 & \frac{\partial}{\partial b_{3=1}} t1 \\ \frac{\partial}{\partial b_{1=1}} t2 & \frac{\partial}{\partial b_{2=1}} t2 & \frac{\partial}{\partial b_{3=1}} t2 \\ \frac{\partial}{\partial b_{1=1}} t3 & \frac{\partial}{\partial b_{2=1}} t3 & \frac{\partial}{\partial b_{3=1}} t3 \end{pmatrix} \right);$$

>

```

M := simplify( eval( Jacobian, {b1 = 1, b2 = 1, b3 = 1} ));
M := [ [ 0.5069592065  0.03203727592  0.03203727592 ]
      [ 0.03203727592  0.5069592065  0.03203727592 ]
      [ 0.03203727592  0.03203727592  0.5069592065 ] ]

```

>

$$x_{fixed} := \begin{bmatrix} 0.2656820273 \\ 0.2656820273 \\ 0.2656820273 \end{bmatrix};$$

$$x_{fixed} := \begin{bmatrix} 0.2656820273 \\ 0.2656820273 \\ 0.2656820273 \end{bmatrix}$$

>

$$q := M. \begin{bmatrix} 0.2656820273 + \epsilon_1 \\ 0.2656820273 + \epsilon_2 \\ 0.2656820273 + \epsilon_3 \end{bmatrix} - M.x_{fixed};$$

$$q := M. \begin{bmatrix} 0.2656820273 + \epsilon_1 \\ 0.2656820273 + \epsilon_2 \\ 0.2656820273 + \epsilon_3 \end{bmatrix} - M.x_{fixed}$$

B.2 Maple Code for the Five-Node Fully-Connected Loop

This code is utilized to find fixed points in the 5 node fully connected system as well as to evaluate the Jacobian of T at a chosen fixed point.

Including only the real domain will eliminate solving for imaginary fixed points.

> with(RealDomain) : with(linalg) :

>

```

P(X1=1, GX2=1) := P(X1=0, GX2=0); P(X1=0, GX3=0) := P(X1=0, GX2=0);
P(X1=1, GX3=1) := P(X1=0, GX2=0); P(X1=1, GX4=1) := P(X1=0, GX2=0);
P(X1=0, GX4=0) := P(X1=0, GX2=0); P(X1=0, GX5=0) := P(X1=0, GX2=0);
P(X1=1, GX5=1) := P(X1=0, GX2=0); P(X2=0, GX1=0) := P(X1=0, GX2=0);
P(X2=1, GX1=1) := P(X1=0, GX2=0); P(X2=0, GX3=0) := P(X1=0, GX2=0);
P(X2=1, GX3=1) := P(X1=0, GX2=0); P(X2=0, GX4=0) := P(X1=0, GX2=0);
P(X2=1, GX4=1) := P(X1=0, GX2=0); P(X2=0, GX5=0) := P(X1=0, GX2=0);
P(X2=1, GX5=1) := P(X1=0, GX2=0); P(X3=0, GX1=0) := P(X1=0, GX2=0);
P(X3=1, GX1=1) := P(X1=0, GX2=0); P(X3=0, GX2=0) := P(X1=0, GX2=0);
P(X3=1, GX2=1) := P(X1=0, GX2=0); P(X3=0, GX4=0) := P(X1=0, GX2=0);
P(X3=1, GX4=1) := P(X1=0, GX2=0); P(X3=0, GX5=0) := P(X1=0, GX2=0);
P(X3=1, GX5=1) := P(X1=0, GX2=0); P(X4=0, GX1=0) := P(X1=0, GX2=0);
P(X4=1, GX1=1) := P(X1=0, GX2=0); P(X4=0, GX2=0) := P(X1=0, GX2=0);
P(X4=1, GX2=1) := P(X1=0, GX2=0); P(X4=0, GX3=0) := P(X1=0, GX2=0);
P(X4=1, GX3=1) := P(X1=0, GX2=0); P(X4=0, GX5=0) := P(X1=0, GX2=0);
P(X4=1, GX5=1) := P(X1=0, GX2=0); P(X5=0, GX1=0) := P(X1=0, GX2=0);
P(X5=1, GX1=1) := P(X1=0, GX2=0); P(X5=0, GX2=0) := P(X1=0, GX2=0);
P(X5=1, GX2=1) := P(X1=0, GX2=0); P(X5=0, GX3=0) := P(X1=0, GX2=0);
P(X5=1, GX3=1) := P(X1=0, GX2=0); P(X5=0, GX4=0) := P(X1=0, GX2=0);
P(X5=1, GX4=1) := P(X1=0, GX2=0);

```

>

$$\begin{aligned} P(Y_{1=1}) &:= \frac{3}{8}; P(Y_{2=1}) := \frac{3}{8}; P(Y_{3=1}) := \frac{3}{8}; P(Y_{4=1}) := \frac{3}{8}; P(Y_{5=1}) := \frac{3}{8}; \\ P(X_{1=1}) &:= \frac{3}{8}; P(X_{2=1}) := \frac{3}{8}; P(X_{3=1}) := \frac{3}{8}; P(X_{4=1}) := \frac{3}{8}; P(X_{5=1}) \\ &:= \frac{3}{8}; P(X_{1=0}GX_{2=1}) := 0.4; P(X_{1=0}GX_{2=0}) := 1 - P(X_{1=0}GX_{2=1}); \end{aligned}$$

>

$$\begin{aligned} P(X_{1=1}GX_{2=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{1=0}GX_{3=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{1=1}GX_{3=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{1=1}GX_{4=1}) := P(X_{1=0}GX_{2=0}); \\ P(X_{1=0}GX_{4=0}) &:= P(X_{1=0}GX_{2=0}); P(X_{1=0}GX_{5=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{1=1}GX_{5=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{2=0}GX_{1=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{2=1}GX_{1=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{2=0}GX_{3=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{2=1}GX_{3=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{2=0}GX_{4=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{2=1}GX_{4=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{2=0}GX_{5=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{2=1}GX_{5=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{3=0}GX_{1=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{3=1}GX_{1=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{3=0}GX_{2=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{3=1}GX_{2=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{3=0}GX_{4=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{3=1}GX_{4=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{3=0}GX_{5=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{3=1}GX_{5=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{4=0}GX_{1=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{4=1}GX_{1=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{4=0}GX_{2=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{4=1}GX_{2=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{4=0}GX_{3=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{4=1}GX_{3=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{4=0}GX_{5=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{4=1}GX_{5=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{5=0}GX_{1=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{5=1}GX_{1=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{5=0}GX_{2=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{5=1}GX_{2=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{5=0}GX_{3=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{5=1}GX_{3=1}) &:= P(X_{1=0}GX_{2=0}); P(X_{5=0}GX_{4=0}) := P(X_{1=0}GX_{2=0}); \\ P(X_{5=1}GX_{4=1}) &:= P(X_{1=0}GX_{2=0}); \end{aligned}$$

>

$$\begin{aligned}
 t5 := & \text{simplify} \left(\text{expand} \left(\left(b_{5=1} \cdot \left(\sum_{i=0}^1 (P(Y_{2=1}) \cdot b_{2=i} \cdot P(X_{5=1}GX_{2=i})) \right) \right) \right. \right. \\
 & \cdot \left(\sum_{j=0}^1 (P(Y_{1=1}) \cdot b_{1=j} \cdot P(X_{5=1}GX_{1=j})) \right) \cdot \left(\sum_{p=0}^1 (P(Y_{3=1}) \cdot b_{3=p} \right. \\
 & \cdot P(X_{5=1}GX_{3=p})) \cdot \left. \left. \left(\sum_{n=0}^1 (P(Y_{4=1}) \cdot b_{4=n} \cdot P(X_{5=1}GX_{4=n})) \right) \right) \right) / \\
 & \left((P(X_{5=1}))^2 \cdot \left(\sum_{k=0}^1 \left(\frac{b_{5=k}}{(P(X_{5=k}))^2} \cdot \left(\sum_{l=0}^1 (P(X_{5=k}GX_{2=l}) \cdot b_{2=l} \cdot P(Y_{2=l})) \right) \right) \right) \right. \\
 & \cdot \left(\sum_{m=0}^1 (P(X_{5=k}GX_{1=m}) \cdot b_{1=m} \cdot P(Y_{1=m})) \right) \cdot \left(\sum_{p=0}^1 (P(Y_{4=1}) \cdot b_{4=p} \right. \\
 & \cdot P(X_{5=k}GX_{4=p})) \cdot \left. \left. \left(\sum_{n=0}^1 (P(Y_{4=1}) \cdot b_{4=n} \cdot P(X_{5=k}GX_{4=n})) \right) \right) \right) \right) \Bigg);
 \end{aligned}$$

$$\begin{aligned}
 t5 := & - (225 \cdot b_{5=1} (16 + 8 \cdot b_{4=1} + 8 \cdot b_{3=1} + 4 \cdot b_{3=1} b_{4=1} + 8 \cdot b_{1=1} + 4 \cdot b_{1=1} b_{4=1} \\
 & + 4 \cdot b_{1=1} b_{3=1} + 2 \cdot b_{1=1} b_{3=1} b_{4=1} + 8 \cdot b_{2=1} + 4 \cdot b_{2=1} b_{4=1} + 4 \cdot b_{2=1} b_{3=1} \\
 & + 2 \cdot b_{2=1} b_{3=1} b_{4=1} + 4 \cdot b_{1=1} b_{2=1} + 2 \cdot b_{1=1} b_{2=1} b_{4=1} + 2 \cdot b_{1=1} b_{2=1} b_{3=1} \\
 & + b_{1=1} b_{2=1} b_{3=1} b_{4=1})) / (-4474 \cdot b_{1=1} b_{2=1} b_{4=1} b_{5=1} \\
 & + 4374 \cdot b_{1=1} b_{2=1} b_{4=1} + 6461 \cdot b_{1=1} b_{2=1} b_{5=1} - 18225 \cdot b_{2=1} b_{1=1} \\
 & b_{4=1}^2 + 8290 \cdot b_{1=1} b_{4=1} b_{5=1} - 965 \cdot b_{5=1} b_{1=1} b_{4=1}^2 + 8290 \cdot b_{2=1} b_{4=1} b_{5=1} \\
 & - 965 \cdot b_{5=1} b_{2=1} b_{4=1}^2 + 12150 \cdot b_{4=1} + 8225 \cdot b_{5=1} - 7290 \cdot b_{2=1} b_{4=1} \\
 & - 7290 \cdot b_{1=1} b_{4=1} - 6561 \cdot b_{1=1} b_{2=1} + 704 \cdot b_{5=1} b_{2=1} b_{1=1} b_{4=1}^2 \\
 & + 10935 \cdot b_{1=1} + 10935 \cdot b_{2=1} - 9935 \cdot b_{2=1} b_{5=1} - 9935 \cdot b_{1=1} b_{5=1} - 2025 \cdot \\
 & b_{4=1}^2 + 1215 \cdot b_{1=1} b_{4=1}^2 + 1215 \cdot b_{2=1} b_{4=1}^2 - 22150 \cdot b_{4=1} b_{5=1} - 475 \cdot b_{5=1} \\
 & b_{4=1}^2)
 \end{aligned}$$

>

$$T := \begin{bmatrix} t1 \\ t2 \\ t3 \\ t4 \\ t5 \end{bmatrix};$$

B.3 Summary of Fixed Points from Maple Results

The following tables summarize fixed points which were found for the three-node fully-connected loop as input parameters were varied.

		$P(X_i=a X_j=b) = 1.0$ where $a=b$	$P(X_i=a X_j=b)=0.0$ where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)			
1	0	Indeterminant			
7/8	1/8	0.5000	0.5000	0.5000	
6/8	2/8	0.5000	0.5000	0.5000	
5/8	3/8	0.5000	0.5000	0.5000	
4/8	4/8	0.5000	0.5000	0.5000	
3/8	5/8	0.5000	0.5000	0.5000	
2/8	6/8	0.5000	0.5000	0.5000	
1/8	7/8	0.5000	0.5000	0.5000	
0	1	indeterminant			

	$P(X_i=a X_j=b)=0.9$, where $a=b$	$P(X_i=a X_j=b)=0.1$, where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	indeterminant		
7/8	1/8	0.7402	0.7402	0.7402
6/8	2/8	0.5469	0.5469	0.5469
5/8	3/8	0.5090	0.5090	0.5090
4/8	4/8	0.5000	0.5000	0.5000
3/8	5/8	NO REAL FIXED POINT		
2/8	6/8	NO REAL FIXED POINT		
1/8	7/8	NO REAL FIXED POINT		
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.8$, where $a=b$	$P(X_i=a X_j=b)=0.2$, where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	Indeterminant		
7/8	1/8	0.8797	0.8797	0.0000
6/8	2/8	0.6082	0.6082	0.6082
5/8	3/8	0.5259	0.5259	0.5259
4/8	4/8	0.5000	0.5000	0.5000
3/8	5/8	NO REAL FIXED POINT		
2/8	6/8	NO REAL FIXED POINT		
1/8	7/8	NO REAL FIXED POINT		
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.7$, where $a=b$	$P(X_i=a X_j=b)=0.3$, where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	Indeterminant		
7/8	1/8	0.8724	0.8724	0.0000
6/8	2/8	0.7672	0.7672	0.0000
5/8	3/8	0.5651	0.5651	0.5651
4/8	4/8	0.5000	0.5000	0.5000
3/8	5/8	NO REAL FIXED POINT		
2/8	6/8	NO REAL FIXED POINT		
1/8	7/8	1.0000	1.0000	1.0000
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.6$, where $a=b$	$P(X_i=a X_j=b)=0.4$, where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	Indeterminant		
7/8	1/8	0.8647	0.8647	0.0000
6/8	2/8	0.7217	0.7217	0.0000
5/8	3/8	0.7304	0.7304	0.7304
4/8	4/8	0.5000	0.5000	0.5000
3/8	5/8	NO REAL FIXED POINT		
2/8	6/8	0.0000	0.0000	0.0000
1/8	7/8	1.0000	1.0000	1.0000
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.5$, where $a=b$	$P(X_i=a X_j=b)=0.5$, where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	Indeterminant		
7/8	1/8	0.8542	0.8542	0.0000
6/8	2/8	0.6250	0.6250	0.0000
5/8	3/8	0.0625	0.0625	0.0000
4/8	4/8	$b_1=1$	$b_2=1$	$b_3=1$
3/8	5/8	0.0000	0.0000	0.0000
2/8	6/8	0.0000	0.0000	0.0000
1/8	7/8	1.0000	1.0000	1.0000
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.4$, where $a=b$	$P(X_i=a X_j=b)=0.6$, where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	Indeterminant		
7/8	1/8	0.0540	0.0540	0.0540
6/8	2/8	NO REAL FIXED POINT		
5/8	3/8	0.2657	0.2657	0.2657
4/8	4/8	0.5000	0.5000	0.5000
3/8	5/8	0.2657	0.2657	0.2657
2/8	6/8	0.0708	0.0708	0.0708
1/8	7/8	0.0117	0.0117	0.0117
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.3,$ where $a=b$	$P(X_i=a X_j=b)=0.7,$ where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	Indeterminant		
7/8	1/8	0.0859	0.0859	0.0859
6/8	2/8	NO REAL FIXED POINT		
5/8	3/8	NO REAL FIXED POINT		
4/8	4/8	0.5000	0.5000	0.5000
3/8	5/8	0.3315	0.3315	0.3315
2/8	6/8	0.0969	0.0969	0.0969
1/8	7/8	0.0169	0.0169	0.0169
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.2,$ where $a=b$	$P(X_i=a X_j=b)=0.8,$ where $a \neq b$		
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)		
1	0	Indeterminant		
7/8	1/8	NO REAL FIXED POINT		
6/8	2/8	NO REAL FIXED POINT		
5/8	3/8	NO REAL FIXED POINT		
4/8	4/8	0.5000	0.5000	0.5000
3/8	5/8	NO REAL FIXED POINT		
2/8	6/8	0.1074	0.1074	0.1074
1/8	7/8	0.0193	0.0193	0.0193
0	1	Indeterminant		

	$P(X_i=a X_j=b)=0.1$, where $a=b$	$P(X_i=a X_j=b)=0.9$, where $a \neq b$			
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)			
1	0	Indeterminant			
7/8	1/8	NO REAL FIXED POINT			
6/8	2/8	0.2649	0.2649	0.2649	
5/8	3/8	0.4139	0.4139	0.4139	
4/8	4/8	0.5000	0.5000	0.5000	
3/8	5/8	0.3372	0.3372	0.3372	
2/8	6/8	0.1109	0.1109	0.1109	
1/8	7/8	0.0202	0.0202	0.0202	
0	1	Indeterminant			

	$P(X_i=a X_j=b)=0.0$, where $a=b$	$P(X_i=a X_j=b)=1.0$, where $a \neq b$			
$P(Y_i=1)=P(X_i=1)$	$P(Y_i=0)=P(X_i=0)$	fixed point coordinates (x,y,z)			
1	0	Indeterminant			
7/8	1/8	0.1328	0.1328	0.1328	
6/8	2/8	0.2787	0.2787	0.2787	
5/8	3/8	0.4207	0.4207	0.4207	
4/8	4/8	0.5000	0.5000	0.5000	
3/8	5/8	0.3377	0.3377	0.3377	
2/8	6/8	0.1109	0.1109	0.1109	
1/8	7/8	0.0204	0.0204	0.0204	
0	1	Indeterminant			

C APPENDIX - MATLAB Simulations and Maple analysis Code

C.1 Markov network code

A simulation was created for the three node Markov network and its derived update function, as seen in equation 4.1. The simulation was used to update the beliefs, P_{xt} , given the past beliefs, P_{xt_1} . Similarly, equation 4.1 was used for the simulation for the behaviour of the tree-structured network. Initial beliefs were only dependent on observation. Dependence on neighbouring nodes, $P_{xi|Givenxi_1}$, was varied to examine the algorithm's behaviour under parameter changes. The code can be easily modified for varying sizes of graphs and input parameters.

C.1.1 Markov network code

The following is the MATLAB code which was used in order to illustrate the belief propagation algorithm's properties under the three node Markov network in Figure 2.

```
%Start Markov network code%
%
%X1-->X2-->X3
%|   |   |
%Y1  Y2  Y3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Notes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%
%Initalization
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
size=3;                %size of markov chain
answer=0;
Pxt=ones(size,2)*1/2;  %P(x)t= [p(Xi=0) P(Xi=1)]
Pxt_1=ones(size,2)*1/2; %P(x)t-1= [p(Xi=0) P(Xi=1)]
PxiGivenxi_1 =ones(size,4);
%P(Xi|Xi-1)=[P(Xi=0|Xi-1=0) P(Xi=1|Xi-1=0) P(Xi=0|Xi-1=1) P(Xi=1|Xi-1=1)]
Px=ones(size,2)*1/2;  %P(x)= [p(Xi=0) P(Xi=1)]
Pxo=ones(size,2)*1/2; %P(Xi|Yi)=[P(Xi=0|Yi) P(Xi=1|Yi)]
Py=ones(size,1)*1/2;
y=[1;0;1];
%y=ones(size,1);      %y=colum vector of randomly generated observation

for(j=1:size)
    y(j,1)=round(rand);
    if (y(j)==1)
        Pxo(j,2)=.75;
```

```

        Pxo(j,1)=.25;
    else
        Pxo(j,1)=.75;
        Pxo(j,2)=.25;
    end
    Pxt_1(j,2)=Pxo(j,2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%loop
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(k=1:10)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Forwards part
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(j=2:size)
    if (round(Pxo(j,2))==round(Pxo(j-1,2)))
        PxiGivenxi_1(j,1)=.75;
        PxiGivenxi_1(j,2)=.25;
        PxiGivenxi_1(j,3)=.25;
        PxiGivenxi_1(j,4)=.75;
    else
        PxiGivenxi_1(j,1)=.25;
        PxiGivenxi_1(j,2)=.75;
        PxiGivenxi_1(j,3)=.75;
        PxiGivenxi_1(j,4)=.25;
    end
end
Pxt(1,2)=Pxt_1(1,2);
for(i=2:size)
    top=Pxo(i,2)*(Pxt_1(i-1,1)*PxiGivenxi_1(i,4)+(Pxt_1(i-1,2)*PxiGivenxi_1(i,2)));
    Pyigiveny=(Py(i)*((Pxo(i,2)/Px(i,2))*((PxiGivenxi_1(i,4)*Pxt_1(i-1,2))+(PxiGivenxi_1(i,2)
        *Pxt_1(i-1,1))))+(((Pxo(i,1))/Px(i,1))*((PxiGivenxi_1(i,3)*Pxt_1(i-1,2))
        +(PxiGivenxi_1(i,1)*Pxt_1(i-1,1))))));
    Pxt(i,2)=top/Pyigiveny;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%update-forwards
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(j=1:size)
    Pxt_1(j,2)=Pxt(j,2); %
    Pxt_1(j,1)=1-Pxt(j,2); %update p(x) at t-1

    if (y(j)==1) %update P(xi|yi)
        Pxo(j,2)=.75;
        Pxo(j,1)=.25;
    else
        Pxo(j,1)=.75;
        Pxo(j,2)=.25;
    end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Backwards part
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(j=1:size-1)
    if (round(Pxo(j,2))==round(Pxo(j+1,2)))
        PxiGivenxi_1(j,1)=.75;
        PxiGivenxi_1(j,2)=.25;
        PxiGivenxi_1(j,3)=.25;
        PxiGivenxi_1(j,4)=.75;
    else
        PxiGivenxi_1(j,1)=.25;
        PxiGivenxi_1(j,2)=.75;
        PxiGivenxi_1(j,3)=.75;
        PxiGivenxi_1(j,4)=.25;
    end
end

Pxt(size,2)=Pxt_1(size,2);
i=size;
while(i>1)
    i=i-1;
    top=Pxo(i,2)*(Pxt_1(i+1,1)*PxiGivenxi_1(i,4)+(Pxt_1(i+1,2)*PxiGivenxi_1(i,2));
    Pyigiveny=(Py(i)*((Pxo(i,2)/Px(i,2))*((PxiGivenxi_1(i,4)*Pxt_1(i+1,2))
    +(PxiGivenxi_1(i,2)*Pxt_1(i+1,1))))+(((Pxo(i,1))/Px(i,1))
    *((PxiGivenxi_1(i,3)*Pxt_1(i+1,2)+(PxiGivenxi_1(i,1)*Pxt_1(i+1,1))))));
    Pxt(i,2)=top/Pyigiveny;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%update-backwards
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(j=1:size)
    Pxt_1(j,2)=Pxt(j,2); %
    Pxt_1(j,1)=1-Pxt(j,2); %update p(x) at t-1

    if (y(j)==1) %update P(xi|yi)?????
        Pxo(j,2)=.75;
        Pxo(j,1)=.25;
    else
        Pxo(j,1)=.75;
        Pxo(j,2)=.25;
    end
end
Pxt_1
end
y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end Markov network code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

C.1.2 Tree-Structured network code

The following is the MATLAB code which was used in order to illustrate the belief propagation algorithm's properties under the tree-structured network seen in Figure 4.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% start Tree network code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Y1
%   |
%  X1
% /  \
% X2  X2
% |   |
% Y2  Y3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Notes
%-insert bottem and test
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%
%Initalization
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
size=7;           %size of markov chain
Pxt=ones(size,2)*1/2; %P(x)t= [p(Xi=0) P(Xi=1)]
Gragh=zeros(size,size); %|0 1 1|-one is added if they are directly conected
                    %|1 0 0|
                    %|1 0 0|
Pxt_1=ones(size,2)*1/2; %P(x)t-1= [p(Xi=0) P(Xi=1)]
PxiGivenxi_1 =ones(size,4);
  %P(Xi|Xi-1)= [P(Xi=0|Xi-1=0) P(Xi=1|Xi-1=0) P(Xi=0|Xi-1=1) P(Xi=1|Xi-1=1)]
Px=ones(size,2)*1/2; %P(x)= [p(Xi=0) P(Xi=1)]
Pxo=ones(size,2)*1/2; %P(Xi|Yi)=[P(Xi=0|Yi) P(Xi=1|Yi)]
Py=ones(size,1)*1/2;
%y=[1;1;0];
y=ones(size,1); %y=column vector of randomly generated observation
top=1;
bottem=1;
for(j=1:size)
  y(j,1)=round(rand);
  if (y(j)==1)
    Pxo(j,2)=.75;
    Pxo(j,1)=.25;
  else
    Pxo(j,1)=.75;
    Pxo(j,2)=.25;
  end
  Pxt_1(j,2)=Pxo(j,2);
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initialize graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
Gragh=[0 1 1 0 0 0 0; 1 0 0 1 1 0 0 ;1 0 0 0 0 1 1;
0 1 0 0 0 0 0;0 1 0 0 0 0 0;0 0 1 0 0 0 0; 0 0 1 0 0 0 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%going up the tree
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%loop-up(to X1)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(k=1:10)
    for(x=1:size)
        test=0;
        top=1;
        bottemx1=1;
        bottemx0=1;
        for(i=1:size)
            if (round(Pxo(x,2))==round(Pxo(i,2)))
                PxiGivenxi_1(i,1)=.75;
                PxiGivenxi_1(i,2)=.25;
                PxiGivenxi_1(i,3)=.25;
                PxiGivenxi_1(i,4)=.75;
            else
                PxiGivenxi_1(i,1)=.25;
                PxiGivenxi_1(i,2)=.75;
                PxiGivenxi_1(i,3)=.75;
                PxiGivenxi_1(i,4)=.25;
            end
            if(Gragh(x,i)==1 && x<i)
                top=top*(Px(i,2)/Py(i))*((Pxt_1(i,2)*PxiGivenxi_1(i,4))+(Pxt_1(i,1)*PxiGivenxi_1(i,3)));
                bottemx0=bottemx0*(Px(i,1))*(PxiGivenxi_1(i,1)*Pxt_1(i,1))
                    +(Px(i,2))*(PxiGivenxi_1(i,2)*Pxt_1(i,2));
                bottemx1=bottemx1*(Px(i,1))*(PxiGivenxi_1(i,3)*Pxt_1(i,1))
                    +(Px(i,2))*(PxiGivenxi_1(i,4)*Pxt_1(i,2));
                test=1;
            end
        end
        if (test==0)
            Pxt(x,2)=Pxt_1(x,2);
        else
            top=Pxo(x,2)*top;
            bottemx0=(Pxo(x,1)/Px(x,1)^2)*bottemx0;
            bottemx1=(Pxo(x,2)/Px(x,2)^2)*bottemx1;
            bottem=bottemx1+bottemx0;
            Pxt(x,2)=top/bottem;
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%update-up
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(j=1:size)
    Pxt_1(j,2)=Pxt(j,2); %
end

```

```

Pxt_1(j,1)=1-Pxt(j,2); %update p(x) at t-1
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Going down the tree
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%loop-down
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=1;
while(x<size+1)
    test=0;
    top=1;
    bottemx1=1;
    bottemx0=1;
    i=size;
    while(i>0)
        if (round(Pxo(x,2))==round(Pxo(i,2)))
            PxiGivenxi_1(i,1)=.75;
            PxiGivenxi_1(i,2)=.25;
            PxiGivenxi_1(i,3)=.25;
            PxiGivenxi_1(i,4)=.75;
        else
            PxiGivenxi_1(i,1)=.25;
            PxiGivenxi_1(i,2)=.75;
            PxiGivenxi_1(i,3)=.75;
            PxiGivenxi_1(i,4)=.25;
        end
        if(Gragh(x,i)==1 && x>i)
            top=Pxo(x,2)*(Px(i)/Py(i))*((Pxt_1(i,2)*PxiGivenxi_1(i,4)
            +(Pxt_1(i,1)*PxiGivenxi_1(i,2)));
            bottem=(Py(i)*(Pxo(x,2)/Px(x,2))*((PxiGivenxi_1(i,4)*Pxt_1(i,2)
            +(PxiGivenxi_1(i,2)*Pxt_1(i,1)))
            +(((Pxo(x,1))/Px(x,1))*((PxiGivenxi_1(i,3)*Pxt_1(i,2)
            +(PxiGivenxi_1(i,1)*Pxt_1(i,1))))));
            i=0;

            Pxt(x,2)=top/bottem;
        else
            i=i-1;
        end
    end
    x=x+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%update-down
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(j=1:size)
    Pxt_1(j,2)=Pxt(j,2); %
    Pxt_1(j,1)=1-Pxt(j,2); %update p(x) at t-1
end

```

```

        if (y(j)==1)                %update P(xi|yi)
            Pxo(j,2)=.75;
            Pxo(j,1)=.25;
        else
            Pxo(j,1)=.75;
            Pxo(j,2)=.25;
        end
    end

Pxt_1
end
y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end Tree network code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

C.2 Fully Connected graph network code

The following is the MATLAB code which was used in order to illustrate the belief propagation algorithm's properties under the fully-connected network seen in Figure 5.

The simulation for the fully connected network was approached in the same way as the Markov network and the tree network, using equation - for the update. The major difference between the previous two simulations and that of the fully-connected network was the method of initializing beliefs. This was done manually for the fully-connected network, whereas beliefs were automatically initialized in the Markov and tree-structure simulations.

```

% Y1
% |
% X1
% / \
% X2---X2
% | |
% Y2 Y3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Notes
%-
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%
%Initialization
%
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
size=5; %size of markov chain
Gragh=zeros(size,size); %|0 1 1|-one is added if they are directly conected
%|1 0 1|
startpoint=.9 %|1 1 0|
Px=ones(size,2)*7/8; %P(x)= [p(Xi=0) P(Xi=1)]
Py=ones(size,1)*7/8; %P(y)= [p(Yi=0) P(Yi=1)]
y=ones(size,1); %y=column vector of randomly generated observation
Pxt=ones(size,2)*startpoint; %P(x|Y,Y,Y)t= [p(Xi=0|Y,Y,Y) P(Xi=1|y,y,y)]
%P(x|Y,Y,Y)t-1= [p(Xi=0|y,y,y) P(Xi=1|y,y,y)]

Pxt_1=ones(size,2)*startpoint;
PxiGivenxi_1 =ones(size,4); %P(Xi|Xi-1)=|P(Xi=0|Xi-1=0) P(Xi=1|Xi-1=0) P(Xi=0|Xi-1=1) P(Xi=1|Xi-1=1)]
% |P(Xi=0|Xi-2=0) P(Xi=1|Xi-2=0) P(Xi=0|Xi-2=1) P(Xi=1|Xi-2=1)]

top=1; %top=top of update equation
bottemx0=1; %bottemx0=bottem half of denominaior where x=0
bottemx1=1; %bottemx1=bottem half of denominaior where x=1
for(j=1:size)
    %y(j,1)=round(rand);
    Px(j,1)=1-Px(j,2);
    Pxt(j,1)=1-startpoint;
    Pxt_1(j,1)=1-startpoint;
    if(y(j,1)==0)
        Py(j,1)=Px(j,1);
    end
end

end
Pxt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initialize gragh
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Gragh=[0 1 1 1 1; 1 0 1 1 1 ;1 1 0 1 1;1 1 1 0 1;1 1 1 1 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Update P(x|x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for (k=1:15)
    for(x=1:size)
        top=1;
        bottemx1=1;
        bottemx0=1;

        for(i=1:size)
            if (round(Pxt_1(x,2))==round(Pxt_1(i,2))) %findP(x|xi)
                PxiGivenxi_1(i,1)=.6;
                PxiGivenxi_1(i,2)=.4;
                PxiGivenxi_1(i,3)=.4;
                PxiGivenxi_1(i,4)=.6;
            else
                PxiGivenxi_1(i,1)=.4;
                PxiGivenxi_1(i,2)=.6;
                PxiGivenxi_1(i,3)=.6;
                PxiGivenxi_1(i,4)=.4;
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% P(Xt|Y,Y,Y)=TP(Xt_1|Y,Y,Y)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (Gragh(x,i)==1)
    top=top*(Py(i))*((Pxt_1(i,2)*PxiGivenxi_1(i,4))+(Pxt_1(i,1)*PxiGivenxi_1(i,3)));
    bottemx0=bottemx0*(Px(i,1))*(PxiGivenxi_1(i,1)*Pxt_1(i,1))+Px(i,2)
    *(PxiGivenxi_1(i,2)*Pxt_1(i,2));
    bottemx1=bottemx1*(Px(i,1))*(PxiGivenxi_1(i,3)*Pxt_1(i,1))+Px(i,2)
    *(PxiGivenxi_1(i,4)*Pxt_1(i,2));
end
end
top=(Pxt(x,2))*top;
bottemx0=(Pxt_1(x,1)/Px(x,1)^2)*bottemx0;
bottemx1=(Pxt_1(x,2)/Px(x,2)^2)*bottemx1;
bottem=(Px(x,2)^2)*(bottemx1+bottemx0);
Pxt(x,2)=top/bottem;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% P(Xt-1)=P(Xt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for(j=1:size)
    Pxt_1(j,2)=Pxt(j,2); %update p(x=1|y,y,y) at t-1
    Pxt_1(j,1)=1-Pxt(j,2); %update p(x=0|y,y,y) at t-1
end

Pxt_1
end
y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%end iteration loop

```